UDC 004.056.55

HIDING DATA IN THE FILE STRUCTURE

A. Kuznetsov, K. Shekhanin, A. Kolgatin, K. Kuznetsova, Ye. Demenko

V. N. Karazin Kharkiv National University, 4 Svobody Sq., Kharkiv, 61022, Ukraine
kuznetsov@karazin.ua, kyryl.shekhanin@nure.ua, kolgatin-a@yandex.ua, kate.kuznetsova.2000@gmail.com, demenjay@gmail.com

Reviewer: Serhii Toliupa, Doctor of Sciences (Engineering), Full Prof., Taras Shevchenko National University of Kiev, 81 Lomonosova St., Kyiv, 03189, Ukraine.
tolupa@i.ua

Received on March 2018

Abstract. *In this paper, the methods of steganography hiding of information in a file system structure is investigated. Namely, the structure of the FAT file system (File Allocation Table) and methods of hiding information messages, which are based on repositioning separate clusters of cover files. A new method is proposed that, unlike the known ones, changes the order of alternation of clusters in each cover file, which allows to further hide a certain informational message, that is, to increase the capacity of the hidden channel. It was confirmed that the results of the data concealment and deletion procedures largely depend on the number of clusters with which it is necessary to carry out the appropriate transformations. It is noted that the extraction procedure is performed much faster than hiding the message. The proposed method is implemented programmatically, the results of experimental researches confirmed the adequacy of the theoretical conclusions and recommendations.*

Keywords*: steganography; hiding information data; file system.*

1 Introduction

Steganographic methods of information protection become, in recent years, increasingly popular and widespread [1-2]. In particular, this is due to the emergence of the latest technologies of hidden communication messages in artificially created containers, redundancy in which is generated by technical features of storage, processing and/or transmission of digital data [3-14]. Namely, methods of network steganography as a carrier (container) use transmitted over the network packet or a set of data packets [3-6]. In the 3D steganography, informational messages hide into artificial excess of digital 3D object models, for example, in the retina of surfaces, holograms, etc. [7-9]. The construction of hidden cluster channels is based on the use of data storage features in modern file systems [10-14]. The last direction is researched in this paper, in detail, researched of methods of steganography hiding of information in the file system structure.

2 Modern file systems

The file system is the procedure established, which determines the way of organizing, storing and naming data on the storage media in computer systems, as well as in other electronic equipment: digital cameras, mobile phones, etc. [15-17]. The file system determines the format of the content and the method of physical storage of information, which is grouped into files. The specific file system defines the size of file names and (directories), the maximum possible file size, and defines the set of file attributes. Some file systems provide service capabilities such as access control or file encryption.

The main functions of the file system are aimed at solving the following tasks: naming files; application file interface; displaying the logical model of the file system on the physical organization of the data warehouse; organization of file system stability to power failure, hardware and software errors; content of the file parameters necessary for its proper interaction with other system objects (kernel, application, etc.).

In multi-user systems, there is another task: protection of one user's files from unauthorized access of another user, as well as collaborative work with files, for example, when a file is opened by one user, for others, the same file will temporarily be available in read-only mode.

The greatest development in computing technology has traditionally been disk drives, the structure of which is generally presented in Fig. 2. Data on disk drives are recorded on tracks. The set of tracks is divided into geometric sectors, while part of the path of a specific geometric sector is called the track sector. The main logical unit of data storage in the file allocation table for disk file systems is a cluster.

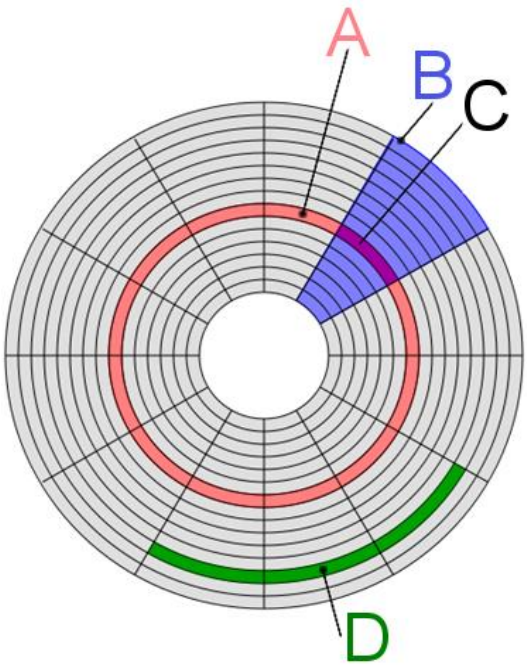

Fig. 1 – Structure of disc:
(A) - track; (B) - geometric sector;
(C) - sector of the track; (D) - cluster.

The cluster is a logical storage unit in a file allocation table that combines a group of sectors of a track. For example, on the 256-byte-sized sector, a 256-byte cluster contains one sector, while the 2-kilobyte cluster contains eight sectors. As a rule, a cluster is the smallest disk space that can be allocated to store a file.

The concept cluster is used in file systems, FAT, NTFS, and others. Other file systems use similar concepts (zones in the Minix, blocks in the Unix, etc.). On some Linux file systems (ReiserFS, Reiser4, Btrfs), BSD (FreeBSD UFS2), the last block of a file can be divided into subfragments, which can be placed "tails" of other files. In NTFS, small files can be written to the Master File Table (*MFT*). The small cluster is best suited for small files. So, this way is more economical. A large cluster allows you to achieve higher speeds, but in small files, the place will be used irrationally (*many sectors will not be fully filled up but will be considered busy*).

Disk file systems are usually stream-oriented. Files in stream-oriented file systems are a sequence of bits, often providing functions such as read, write, change data and control access. The most common current stream-oriented file systems are FAT (*File Allocation Table*) - its three different types (*FAT16\32\64*) and NTFS (*New Technology File System*). Given the complete openness of the file system specification in this paper, only FAT is considered below.

The structure of the FAT consists of five parts: Volume ID; FAT - tables (*two examples*); Clusters (*data files*); Root directory.

Volume ID located at the beginning of the disk partition of the FAT file system. It is required for the initial boot of the device. It also contains information about the parameters of the file system.

File Allocation Table is intended to indicate clusters of individual files. The disk data area is separated into clusters - blocks that are sized when formatting a disk. Each file and directory occupy one or more clusters. Thus, clusters of chains are formed. In the file allocation table, each cluster is marked in a special way. The pointer size in bits for each cluster is specified in the file system name. For example, for the FAT32 file system, the size of the pointer is 32 bites. There are three types of cluster pointers:

- free cluster is a cluster in which new files and directories will be recorded;.
- busy cluster - the pointer indicates the next cluster in the chain. If the chain of clusters is over, then the cluster is marked with a special value (0xFFFFFF0F in hex);
- Bad block - cluster with access errors. Indicated when formatting the drive to disable later access to it.

Damage to the file allocation table completely destroys the file system structure, so two copies of the table are always stored on the disk.

Clusters (data files) - the data area that is placed directly after the last FAT table. The FAT directory (folder, directory) is an ordinary file marked with a special attribute. The data of such a file in any version of FAT is a chain of 32-byte file records (*directory entries*). The catalog cannot contain two files with the same names. If the disk validation program detects an artificially created pair of files with the same name in one directory, one of them is renamed.

Root directory – the disk area in which the root directory information is located. Its size is limited, so in the root directory of the disk can be no more than 512 files and subdirectories.

The main advantage of the FAT file system is its simplicity and compatibility with outdated operating systems. For this file system, there is a many detailed open documentations. A breach in the system often lead to damage to one or more files. However, in case of serious damage, it is much easier to restore information than NTFS.

3 Steganographic methods of hiding information in the file system structure

The simplest steganographic methods of hiding information in the structure of the file system are discussed in [10,11]. They use free clusters (or certain service data fields) to record a hidden message, but this method is unreliable [13,14]. Other methods, such as [12-14], are based on the use of multiple cover files and hiding an informational message by changing the relative positions of the clusters of different cover files one to another.

The hidden data is presented in the form of a bit array:

$$M = \{b_0, b_1, ..., b_{n-1}\},\ b_i \in \{0,1\}.$$

On the device $p = 2^m$, $m \in N$ cover files are selected:

$$F_0, F_1, ..., F_{p-1}.$$

The order of the clustering of the cover files will hide the information message, that is, after the embedding, the cover files cannot be deleted, moved or modified. The natural number m and names of the cover files is the secret key. Also important is the order of cover files [14].

An array of cluster numbers for cover files is formed:

$$C = \begin{pmatrix} c_{0,0} & c_{0,1} & \dots & c_{0,L_0-1} & & & \\ c_{1,0} & c_{1,1} & \dots & \dots & \dots & \dots & c_{1,L_1-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p-1,0} & c_{p-1,1} & \dots & \dots & c_{p-1,L_{p-1}-1} & & \end{pmatrix}, \quad (1)$$

where each row of the array contains the cluster numbers of the corresponding file. For example, the file F_i corresponds to the i line of the array C, that is, the cluster numbers of the i covering file can be represented as an array

$$C_{F_i} = \{c_{i,0}, c_{i,1}, ..., c_{i,L_i-1}\}$$

where L_i - clusters number in i cover file.

If, when hiding the information, it is necessary to save without changing the contents of the cover files, then it is necessary that the conditions are fulfilled: $\forall i : L_i \geq k$, $k = n/m$. An array D of empty file system cluster numbers is formed: $D = \{c_1, c_2, ..., c_{L_D}\}$, And $c_1 < c_2 < ... < c_{L_D}$.

The number L_D is the number of empty clusters of the file system, and it is required that the condition is fulfilled:

$$L_D \geq \sum_{i=0}^{p-1} L_i.$$

Information message M is separated into blocks by m bits each: $M = \{B_1, B_2, ..., B_k\}$, that $k = \lceil n/m \rceil$ and if $k = n/m$, then $B_1 = \{b_0, b_1, ..., b_{m-1}\}$, $B_2 = \{b_m, b_{m+1}, ..., b_{2m-1}\}$, …,

$B_k = \{b_{(k-1)m}, b_{(k-1)(m+1)}, ..., b_{km-1}\}$. If $k < n/m$, then the last block is supplemented by zeros $B_1 = \{b_0, b_1, ..., b_{m-1}\}$, $B_2 = \{b_m, b_{m+1}, ..., b_{2m-1}\}$, …, $B_k = \{b_{(k-1)m}, b_{(k-1)(m+1)}, ..., b_{n-1}, \underbrace{0, 0, ..., 0}_{km-n}\}$.

Each block B_i, $i = 1, 2, ..., k$ is interpreted to the natural number $\forall i : 0 \le B_i \le p-1$. Each of the natural number. B_i., $i = 1, 2, ..., k$ is interpreted to the number of cover files from files array $F_0, F_1, ..., F_{p-1}$.

All clusters of the cover files are overwritten to empty clusters, that is, the D array is filled with cluster numbers from the array C. The order of overwriting clusters of cover files corresponds to a sequence of natural numbers $\{B_1, B_2, ..., B_k\}$, which are conditioned by the information message. For example, the first empty cluster overwrites the first cluster of the cover file with number B_1, in the second empty cluster overwrites the next cluster of the cover file with the number B_2 etc. The natural numbers B_i may coincide, and in this case, the clusters of the same cover file with the number B_i are written. In order to enhance the protection, various techniques can be used, for example [14], the initial value of B_0, is selected, and the order of overwriting the clusters of the covering files is given by a sequence of positive integers $\{N_1, N_2, ..., N_k\}$,

$$N_i = B_{i-1} + B_i \bmod p,\ 0 \le N_i \le p-1.$$

After, the first empty cluster overwrites the first cluster of the cover file with the number N_1, in the second empty cluster - the next cluster of the cover file with the number N_2 etc. At the result of the algorithm, the first k empty clusters of the file system will be written by clusters of the cover files. So the condition $k \le L_D$ must be fulfilled.

To extract the information message M the array D of the clusters of the cover files is formed: $D = \{c_1, c_2, ..., c_{L_D}\}$, and $c_1 < c_2 < ... < c_{L_D}$. Each cluster number in this array is correlated with only one cluster of the cover file. This correspondence is determined by the logic of embedding information and is used to extract data. In this case, a sequence of natural numbers is formed $\{B_1, B_2, ..., B_k\}$, which correspond to the blocks of the informational message:

$$B_1 = \{b_0, b_1, ..., b_{m-1}\},\ B_2 = \{b_m, b_{m+1}, ..., b_{2m-1}\},\ ...,\ B_k = \{b_{(k-1)m}, b_{(k-1)(m+1)}, ..., b_{km-1}\}.$$

The informative message is calculated from these bit blocks

$$M = \{b_0, b_1, ..., b_{n-1}\},\ b_i \in \{0,1\}.$$

If $k < n/m$, then the last block is "cut" - its last $km-n$ bits are not used.

The disadvantage of this method is a small size of hidden data, which depends on the number of cover files and the size of the one cluster at the file system. Each cluster of cover files can contain $\log_2 p = m$ information bits. In this paper, we propose a new steganographic method of hiding information in the structure of the file system, which, in contrast to the one discussed, further modifies the order of the cluster alternation in each cover file. The improved method allows an increase in the amount of hidden information is achieved.

4 Proposed method

The proposed method of steganographic hiding of data in cluster file systems is based on the use of several cover files (*as in the prototype method*) and the hiding of a secret message by changing the relative positions of clusters of different cover files one to the other and, unlike the known methods, the order of alternating clusters in each cover file. The hidden data is represented as the bit array:

$$M = \{b_0^*, b_1^*, ..., b_{L_1+L_2+...+L_{p-1}-1}^*, b_0, b_1, ..., b_{n-1}\},\ b_i^*, b_i \in \{0,1\}.$$

On the device $p = 2^m$, $m \in N$ cover files are selected: $F_0, F_1, ..., F_{p-1}$.

The array of cluster numbers of cover files is formed as (1). For each cover file, the order of alternating clusters in each cover file is changed. The order of alternation is given by the information sequence M. To do this, p bit arrays of information bits are formed

$$M_1 = \{b^*_0, b^*_1, ..., b^*_{L_1-1}\},$$
$$M_2 = \{b^*_{L_1}, b^*_{L_1+1}, ..., b^*_{L_1+L_2-1}\},$$
$$\ldots,$$
$$M_{L_p} = \{b^*_{L_1+L_2+...+L_{p-1}}, b^*_{L_1+L_2+...+L_{p-1}+1}, ..., b^*_{L_1+L_2+...+L_{p-1}-1}\},$$

each of which is mapped to an array of cluster numbers of files

$$C_{F_1} = \{c_{1,0}, c_{1,1}, ..., c_{1,L_1-1}\},$$
$$C_{F_2} = \{c_{2,0}, c_{2,1}, ..., c_{2,L_2-1}\},$$
$$\ldots,$$
$$C_{F_p} = \{c_{p,0}, c_{p,1}, ..., c_{p,L_p-1}\}.$$

The clusters of each cover file are reordered, that is, the cluster numbers in each of C_{F_1}, C_{F_2}, … , C_{F_p} arrays change their alternation in accordance with the values of the bit arrays M_1, M_2, … , M_{L_p}. As a result, new arrays of cluster numbers are obtained $C^*_{F_1}$, $C^*_{F_2}$, … , $C^*_{F_p}$. Reordering clusters in each cover file can be done in different ways. For example, by splitting down all numbers $\{c_{i,0}, c_{i,1}, ..., c_{i,L_i-1}\}$ into two halves and comparing each half with the value of the information bit $b^*_j = 1$, $L_1 + L_2 + ... + L_{i-1} - 1 < j \le L_1 + L_2 + ... + L_i - 1$, on a j position in array $C^*_{F_i}$ place a cluster from the first half of ordered numbers, if $b^*_0 = 0$ - from the second half.

Formed in this way arrays $C^*_{F_i} = \{c^*_{i,0}, c^*_{i,1}, ..., c^*_{i,L_i-1}\}$ reordered numbers of cover files form an array of

$$C^* = \begin{pmatrix} c^*_{0,0} & c^*_{0,1} & \ldots & c^*_{0,L_0-1} & & & \\ c^*_{1,0} & c^*_{1,1} & \ldots & \ldots & \ldots & \ldots & c^*_{1,L_1-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ c^*_{p-1,0} & c^*_{p-1,1} & \ldots & \ldots & c^*_{p-1,L_{p-1}-1} & & \end{pmatrix}.$$

Changing the alternation of clusters in each cover file allows you to hide the first $L_1 + L_2 + ... + L_{p-1}$ information bits from the array M, that is, the information sequence $\{b^*_0, b^*_1, ..., b^*_{L_1+L_2+...+L_{p-1}-1}\}$. The rest of the information bits are hidden in the same way as in the above prototype method [14].

The array D of empty file system clusters is formed: $D = \{c_1, c_2, ..., c_{L_D}\}$, $c_1 < c_2 < ... < c_{L_D}$. The sequence of information bits $\{b_0, b_1, ..., b_{n-1}\}$ is separated into blocks by m bits each: $\{B_1, B_2, ..., B_k\}$, Each block B_i, $i = 1, 2, ..., k$ s interpreted as a natural number, i.e. $\forall i : 0 \le B_i \le p - 1$. Each natural number B_i, $i = 1, 2, ..., k$ is interpreted as the number of the cover file from the set of files $F_0, F_1, ..., F_{p-1}$. All cluster of the cover files are overwritten in empty clusters, that is, the array D is filled with cluster numbers from the array C^* (reordered clusters, that is, with the alternating cluster interchanges in each cover file). The order of rewrite of cluster covers files corresponds to a sequence of natural numbers $\{B_1, B_2, ..., B_k\}$, that are specified by the information message. For example, the first empty cluster overwrites the first cluster of the cover file with number B_1, in the second empty cluster, the next cluster of the cover file with the number B_2

etc. The natural numbers B_i may coincide, and in this case, the regular clusters of the same cover file with the number B_i are written. As a result, the first k empty clusters of the file system will be recorded by the clusters of the cover files. To extract the information message M, the array D of the cluster numbers of the cover files is formed: $D=\{c_1,c_2,...,c_{L_D}\}$. Each cluster number in this array is correlated with only one cluster of the cover file. In this case, a sequence of natural numbers $\{B_1,B_2,...,B_k\}$, is formed that correspond to the blocks of the informational message, i.e. the information sequence $\{b_0,b_1,...,b_{n-1}\}$, $b_i\in\{0,1\}$ is formed. Then the information sequence is extracted

$$\{b_0^*,b_1^*,...,b_{L_1+L_2+...+L_{p-1}-1}^*\},\ b_i^*\in\{0,1\}.$$

For this purpose, the arrays $C_{F_i}^*=\{c_{i,0}^*,c_{i,1}^*,...,c_{i,L_i-1}^*\}$ of the cluster numbers of each cover file are analyzed. The extraction rule corresponds to the logic of hiding. For example, the splitting of all ordered $\{c_{i,0},c_{i,1},...,c_{i,L_i-1}\}$ numbers into two halves and the matching of each half with the value of the information bit can be applied. For example, if there is a cluster on the j position in array $C_{F_i}^*$ on the first half of an array of ordered numbers $\{c_{i,0},c_{i,1},...,c_{i,L_i-1}\}$, accepts $b_j^*=1$. If the second half accepts $b_0^*=0$. Thus, due to the additional change in the order of alternation of clusters in each cover file, it is possible to increase the size of hidden information. In particular, in comparison with the prototype method, it is possible to hide one bit per cluster of cover files in an additional way.

The proposed method was implemented programmatically, experimental research of its effectiveness was conducted. Fig. 2 and 3 contain results of the comparative analysis of the built-in data capacity by a base method and the method offered in this work are given.

Fig 2 shows the dependence of the size of the steganograms on the size of the cluster of cover files. As it is seen, that is doubling the bandwidth of steganogram.

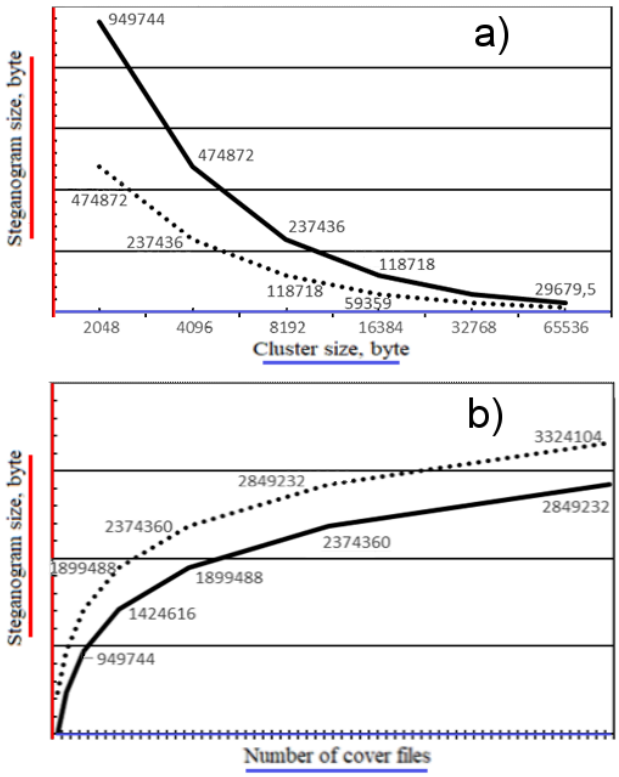

Fig. 2 – Dependence between:
- message length and size of a cluster (a), and count of cover files (b)

In fig. 3 shows the size of the steganograms depending on the number of cover files. There are comparing the results show that with equal input parameters, the proposed method allows hiding twice longer message than the basic method prototype. In addition, the improved method allows you to use only one cover file, as compared to the basic method. By comparing the computational complexity of the methods, it can be argued that the improved method requires twice the computational resources. For experimental research of the effectiveness of the methods, the program "*Stegano FAT*", FAT 32 file system on a JetFlash 350 Transcend® flash drive with a capacity of 8 GB, USB 2.0 connection interface and laptop Lenovo® Y510P with Windows® ver. 8.1 OS were used.

It should be noted that the actual execution time of hiding methods depends on both the hardware features of the data carriers and the algorithmic implementation. We will analyze the operation time of the methods, depending on the selected parameters: size of cluster; size of message; count of cover files; total size of the cover files.

To estimate the dependence of the time spent on the executing of the hiding and extracting messages methods to the cluster size, we set the message size is 100 bytes, the number of cover files – 2, the total size of the cover file – 7 MB.

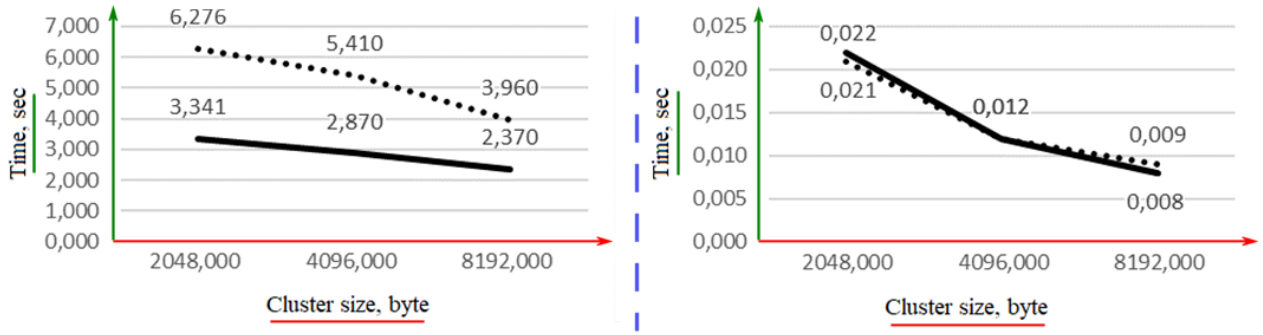

Fig. 3 – Effect of size of a cluster on spent time

We will change the cluster size in the range: 2048, 4096, 8192 bytes. The results of the experiment are summarized in Fig. 3. In this graph and the following ones, on the left are the results of hiding the message, on the right – when extracting. The dotted line shows the results of the modified method, the solid line shows the results of the basic method. As seen in Fig. 3, with increasing cluster size, the time of the concealment of the message is reduced. The improved method requires twice as much time to hide information in contrast to the based method, with the same configuration. The selected method does not affect the time of information retrieval. To estimate the dependence between the spent time on the hiding and extracting message and size of the message, we set the cluster size is 2048 bytes, the number of cover files – 2, the total size of the cover files 7 MB. We will change the size of the message: 100, 200, 400 bytes.

The results of the time spent analysis are shown in Fig. 4. As seen in Fig. 4 as the message size increases, the time to hide and extract the message increases. To estimate the dependence of the time spent on hiding and extracting the message to the number of cover files, we set the size of the cluster - 2048 bytes, the message size is 100 bytes, the total size of the cover files - 7 MB. We will change the number of cover files: 2,4,8. The results are shown in Fig. 5.

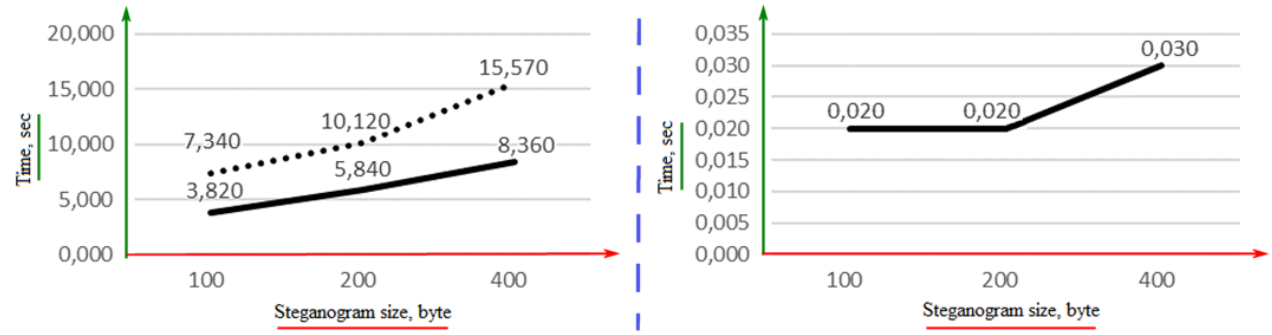

Fig. 4 – Effect of the size of the message on spent time

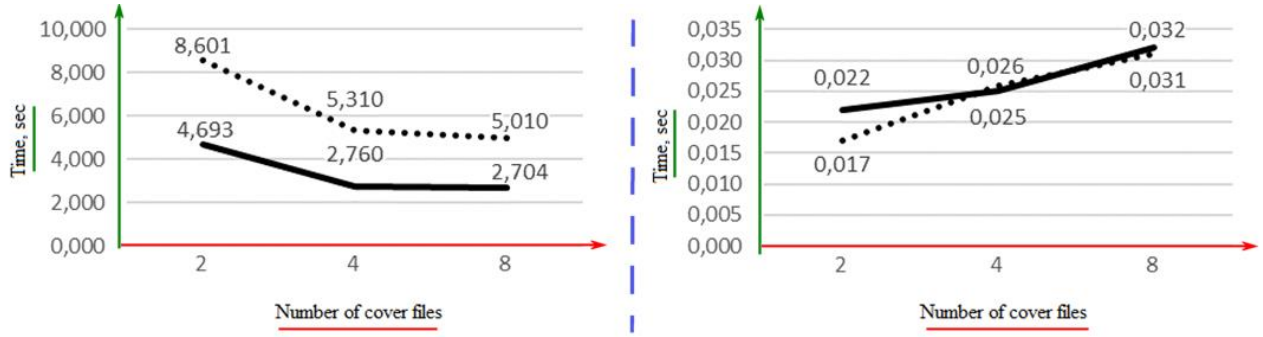

Fig. 5 – Effect of a count of cover files on spent time

As seen in Fig. 5, as the number of cover files increases, the time to hide the message - decreases. This is due to the fact that the number of information clusters, with an increase in the number of cover files, decreases, and accordingly increases the number of clusters that will be recorded without mixed. When a message is extracted, the time spent increases according to the number of cover files. To estimate the dependence of the time spent on the hiding and extracting of the message on the total size of the cover files, we fix the cluster size - 2048 bytes, the number of cover files - 2, size of the message is 100 bytes. We will change the total size of the cover files: 1.7, 3.5, 7 MB. The results are shown in Fig. 6. As seen in Fig. 6, when the total size of the cover files increasing, time to hiding and extracting the message increases.

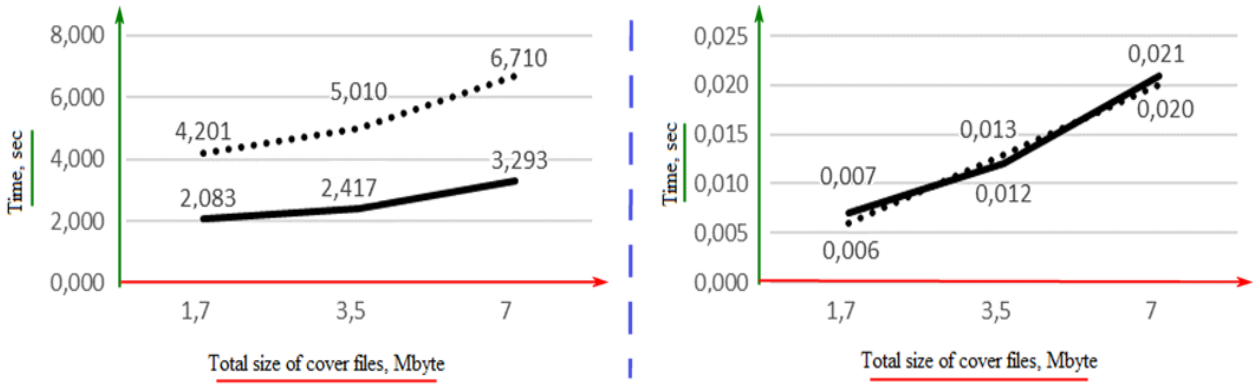

Fig. 6 – Effect of the total size of cover files on spent time

5 Conclusions

In this paper, we propose a new method, which, in contrast to the known besides regrouping clusters covering files additionally changes the order of alternating clusters in each of the cover files. It allows to further hide a specific information message, that is, increase the bandwidth of the hidden channel. The proposed method is implemented programmatically, the results of experimental research confirmed the adequacy of the theoretical conclusions and recommendations. There are results can be argued: the time of concealment and deletion of the message is largely influenced by the number of clusters over which we need to make a reposition; Extracting is performed much faster than concealing the message.

References

1. S. Katzenbeisser, F. A. Petitcolas. Information Hiding Techniques for Steganography and Digital Watermarking, Norwood, MA, USA: Artech House, 2000, 220 p.
2. F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn. "Information hiding-a survey," in Proc. of the IEEE, Vol. 87, No. 7, pp. 1062-1078, Jul 1999.
3. W. Mazurczyk, M. Smolarczyk, K. Szczypiorski. "Retransmission steganography and its detection", Soft Computing, Vol. 15, No. 3, pp. 505-515, 2011.
4. S. Nair, A. Kumar, A. Sur and S. Nandi. "Length based network steganography using UDP protocol". 2011 IEEE 3rd Int. Conference on Communication Software and Networks, Xi'an, 2011, pp. 726-730.
5. K. Ahsan and D. Kundur. "Practical data hiding in TCP lIP", In: ACM Workshop on Multimedia and Security, 2002, [On-line]. Internet: http://ee.tamu.edu/deepalpdf/ acm02.pdf
6. S. H. Sellke, C. Wang, S. Bagchi, and N. B. Shroff. "TCP/IP Timing Channels: Theory to Implementation", pp. 2204-2212, 2009.
7. V. Itier, W. Puech and A. G. Bors. "Cryptanalysis aspects in 3-D watermarking". 2014 IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 4772-4776.
8. Yang, Qin, Liujie, Sun, and Wenju, Wang. "A robust watermarking scheme for 3D models based on encrypted holographic algorithm". Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things, Harbin, 2015, pp. 85-89.
9. Z. Li, S. Beugnon, W. Puech, and A. G. Bors. "Rethinking the high capacity 3D steganography: Increasing its resistance to steganalysis". 2017 IEEE (ICIP), Beijing, 2017, pp. 510-414.
10. S. F. Liu, S. Pei, X. Y. Huang, and L. Tian. "File hiding based on FAT file system". 2009 IEEE International Symposium on IT in Medicine & Education, Jinan, 2009, pp. 1198-1201.

11. J. Davis, J. MacLean and D. Dampier. "Methods of Information Hiding and Detection in File Systems". 2010 Fifth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering, Oakland, CA, 2010, pp. 66-69.
12. H. Khan, M. Javed, S.A. Khayam, F. Mirza. “Designing a cluster-based covert channel to evade disk investigation and forensics”. Computers & Security, Vol. 30, Issue 1, January 2011. [On-line]. Internet: https://www.sciencedirect.com/science/article/pii/S016740481000088X
13. H. Khan, M. Javed, S.A. Khayam, F. Mirza. “Evading Disk Investigation and Forensics using a Cluster-Based Covert Channel”. National University of Science & Technology (NUST), Islamabad 44000, Pakistan. [On-line]. Internet: https://www.sigsac.org/ccs/CCS2009/pd/abstract_17.pdf
14. N. Morkevičius, G. Petraitis, A. Venčkauskas, J. Čeponis. “Covert Channel for Cluster-based File Systems Using Multiple Cover Files”. Information Technology and Control, 2013, Vol. 42, No.3. pp. 32. [On-line]. Internet: http://itc.ktu.lt/index.php/ITC/article/view/3328 .
15. L. Yang, P. Chen, G. Zhu, and L. Yu. "Repairing algorithm design for FAT file system in embedded system". 2011 International Conference on Consumer Electronics, Communications and Networks (CECNet), XianNing, 2011, pp. 3393-3396.
16. Z. Jinhai. "Research of embedded FAT file system". 2011 International Conference on Uncertainty Reasoning and Knowledge Engineering, Bali, 2011, pp. 44-47.
17. H. Zhao, X. Li, L. Chang, and X. Zang, "Fat File System Design and Research". 2015 International Conference on Network and Information Systems for Computers, Wuhan, 2015, pp. 568-571.

Рецензент: Сергій Толюпа, д.т.н., проф., Київський національний університет імені Т. Шевченка, м. Київ, Україна.
E-mail: tolupa@i.ua

Надійшло: Березень 2018.

Автори:
Олександр Кузнецов, д.т.н., проф., академік Академії наук прикладної радіоелектроніки, ХНУ імені В.Н. Каразіна, м. Харків, Україна.
E-mail: kuznetsov@karazin.ua

Кирил Шеханін, аспірант, ХНУ імені В.Н. Каразіна, м. Харків, Україна.
E-mail: kyryl.shekhanin@nure.ua

Андрій Колгатін, студент факультету комп’ютерних наук, Харківський національний університет імені В.Н. Каразіна, м. Харків, Україна.
E-mail: kolgatin-a@yandex.ua

Катерина Кузнецова, студентка факультету комп’ютерних наук, ХНУ імені В.Н. Каразіна, м. Харків, Україна.
E-mail: kate.kuznetsova.2000@gmail.com

Євген Деменко, студент факультету комп’ютерних наук, Харківський національний університет імені В.Н. Каразіна, м. Харків, Україна.
E-mail: demenjay@gmail.com

Приховування даних в файлової структурі.

Анотація. У статті досліджуються методи стеганографії, що приховують інформацію в структурі файлової системи. А саме, структура файлової системи FAT (таблиця розподілу файлів) і методи приховування інформаційних повідомлень, що засновані на зміні положення окремих кластерів файлів обкладинки. Пропонується новий метод, який, на відміну від відомих, змінює порядок слідування кластерів в кожному файлі обкладинки, що дозволяє додатково приховати інформаційне повідомлення, тобто збільшити ємність прихованого каналу. Підтверджено, що результати процедур приховування та вилучення даних в значній мірі залежать від кількості кластерів, з якими необхідно провести відповідні перетворення. Відзначено, що процедура вилучення виконується набагато швидше, ніж приховування повідомлення. Пропонований метод реалізований програмно, а результати експериментальних досліджень підтверджують правильність теоретичних висновків і рекомендацій.

Ключові слова: стеганографія; приховування інформаційних даних; файлова система.

Рецензент: Сергей Толюпа, д.т.н., проф., Киевский национальный университет имени Т. Шевченко, г. Киев, Украина.
E-mail: tolupa@i.ua

Поступила: Март 2018.

Авторы:
Александр Кузнецов, д.т.н., проф., академик Академии наук прикладной радиоэлектроники, ХНУ имени В.Н. Каразина, г. Харьков, Украина.

E-mail: kuznetsov@karazin.ua

Кирилл Шеханин, аспирант, ХНУ имени В. Н. Каразина, г. Харьков, Украина.
E-mail: kyryl.shekhanin@nure.ua

Андрей Колгатин, студент факультета компьютерных наук, ХНУ имени В.Н. Каразина, г. Харьков, Украина.

E-mail: kolgatin-a@yandex.ua

Екатерина Кузнецова, студентка факультета компьютерных наук, ХНУ имени В.Н. Каразина, г. Харьков, Украина.
E-mail: kate.kuznetsova.2000@gmail.com

Евгений Деменко, студент факультета компьютерных наук, ХНУ имени В.Н. Каразина, г. Харьков, Украина.
E-mail: demenjay@gmail.com

Скрытие данных в файловой структуре.

Аннотация. В статье исследуются методы стеганографии, скрывающие информацию в структуре файловой системы. А именно, структура файловой системы FAT (таблица распределения файлов) и методы скрытия информационных сообщений, которые основаны на изменении положения отдельных кластеров файлов обложки. Предлагается новый метод, который, в отличие от известных, изменяет порядок чередования кластеров в каждом файле обложки, что позволяет дополнительно скрыть информационное сообщение, то есть увеличить емкость скрытого канала. Подтверждено, что результаты процедур сокрытия и извлечения данных в значительной степени зависят от количества кластеров, с которыми необходимо провести соответствующие преобразования. Отмечено, что процедура извлечение выполняется гораздо быстрее, чем скрытие сообщения. Предлагаемый метод реализован программно, а результаты экспериментальных исследований подтверждают правильность теоретических выводов и рекомендаций.

Ключевые слова: стеганография; скрытие информационных данных; файловая система.