Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.024494
Article

Tech Science Press

Software Information Hiding Algorithm Based on Palette Icon of PE File

Zuwei Tian[1,*], Hengfu Yang[1] and Zhichen Gao[2]

[1]College of Information Science and Engineering, Hunan First Normal University, Changsha, 410205, China
[2]Department of Applied Mathematics and Statistics, College of Engineering and Applied Sciences, Stony Brook University, Stony Brook, NY, 11794-2300, USA
*Corresponding Author: Zuwei Tian. Email: tianzuwei@126.com
Received: 19 October 2021; Accepted: 19 November 2021

Abstract: PE (Portable executable) file is a standard format for executable file and is applied extensively. PE file has diversity, uncertainty of file size, complexity of file structure and singleness of file format, which make PE file easy to be a carrier of information hiding, especially for that of large hiding capacity. A novel software information hiding algorithm is proposed, which makes full use of display characteristics of palette icon of portable executable file. In this algorithm, the information is embedded into the transparent area of the icon by taking advantage of the redundant color items in the palette. The experimental results show that after embedding the information, the size of the icon remains unchanged, that is, the size of the resource section will not change, and the size of the PE file will not change. On the other hand, the icon with embedded data can be correctly analyzed and displayed without any distortion. PE file can run normally, and does not affect the performance of the program, so the algorithm has good concealment. The algorithm selects an index value whose color is black according to rules in the XOR bitmap, these index values are encoded for information hiding, its complexity is low. At the same time, we can further improve the hiding capacity by adding one or more icons to PE file.

Keywords: Information hiding; portable executable file; palette icon; transparent display

1 Introduction

PE file is a standard Windows executable file format [1], which originates from the version of the Unix COFF (Common Object File Format) file format. A PE file is composed of DOS (Disk Operating System) header, DOS stub, PE header, section table and section, which plays a very important role in the Windows operating system. PE files are widely used in Win32 executable programs, including EXE, DLL, OCX, SYS, SCR and so on.

PE files are widely used in the Windows operating system, however, the related information hiding algorithms are proposed recently. Few research and literature are found in this field till now. The information hiding scheme based on PE file can be divided into the following categories: Firstly, information hiding technique based on instruction transformation is presented [2,3]. The compiler optimized binary codes are directly modified by means of equivalent instruction's replacement, this method is simple, however, it affects the performance of the program, and the hiding capacity is very limited. Secondly, information hiding method based on the PE file redundant space is presented. In [4–6], many redundant spaces in PE file are used to hide information. The method does not affect the performance of program nor increase the file size. Different users can use password encryption method to achieve covert communication. However, it has relatively concentrated hiding space and poor imperceptibility. Fang et al. proposed an information hiding method, in which the information is camouflaged to prevent it from being understood or tampered, by making full use of the redundant space of PE file structure, redundant fields and static spaces allocated to strings [7]. Thirdly, the method of hiding the information is based on the structural features of PE file itself, such as the structure of PE file resources section [8]. This method does not add nor modify the redundant space of PE file, so it is better than the first method in invisibility and robustness. However, the default PE file structures of the resources are ordered, and it will destroy its default order to embed information by changing resource section structure, and arouse suspicion easily.

Duanmu et al. proposed a new information hiding method based on the storage format and features of icon and bitmap resource [9]. The method embeds secret information into PE resources datum by utilizing the imperceptibility of PE file and by applying effective information hiding algorithms. The experiment shows that it can enhance the information imperceptibility and capacity. In addition, a new palette redundant color allocating algorithm is proposed, which can improve the embedding capacity of information hiding algorithms based on palette color redundancy. Disadvantage: change resource section structure, restore the redundant field values of the resource section, rearrange and modify the color palette entries, etc., these will result in failure to extract the hidden information.

Xu et al. proposed an information hiding algorithm based on bitmap resource of portable executable file [10]. This algorithm has higher security than some traditional ones because of integrating secret data and bitmap resources together. Through analyzing the principle of bitmap resources parsing in an operating system and the layer of resource data in PE files, a safe and useful solution is presented to solve two problems that bitmap resources are incorrectly analyzed and other resources data are confused in the process of data embedding. The feasibility and effectiveness of the proposed algorithm are confirmed through computer experiments.

There are some defects in the above information hiding algorithms, such as too concentrated hidden information, poor imperceptibility, not closely related to the functionality of the program, and affecting the compiler optimized program performance. In order to achieve covert communication of PE file without affecting the performance of the program nor increasing the length of the program, this paper proposes a new software information hiding algorithm based on PE file icon resource by deeply analyzing the principle of palette icon transparency display and making full use of the transparent display characteristics of PE file icon resources.

2 Structure of PE File Icon Resource

To make the user's interface more attractive and achieve a better visual effect, PE file contains a large number of multimedia resources as the design materials, including bitmaps, icons, menus, text resources, dialog template, GIF animations, and WAVE sound, and etc. These multimedia software resources are compiled into the binary information and stored in the resource section of PE file. In general, the name of resource section is .rsrc, which organize the datum by using a disk directory tree structure [11,12]. It is divided into 5 layers, namely, the directory of resource type, the directory of resource ID (Identity Document), resources, code page directory of resources, resource description information entry and resource data.

2.1 PE File Icon Resource

Icon resource is an important part of PE file. The explorer can browse the files in various ways, such as thumbnails, tiles, icons, list and detail information. When Windows needs to display the program icon, they will find the most appropriate icon to display it according to display color quality of the current monitor, the current system language, and different viewing ways.

In order to obtain the best effect of icon display in different operating systems and different color depth of desktop, A PE file contains a number of icon images with a variety of sizes and color depths. The Windows operating system will choose the appropriate size/color depth icon from the resource to cater for different context of the icon when the program is running. Generally, in order to get a good display in different environment, application programmer needs to design three sizes icons: 48 × 48 pixels, 32 × 32 pixels, 24 × 24 pixels, 16 × 16 pixels. Each icon should contain these three color depths: 32 bit (24-bit with 8-bit alpha), 8-bit (256 colors), 4-bit (16 colors), see Fig. 1. If the operating system cannot find a particular icon image format, it will display the nearest icon. It works but quality suffers.

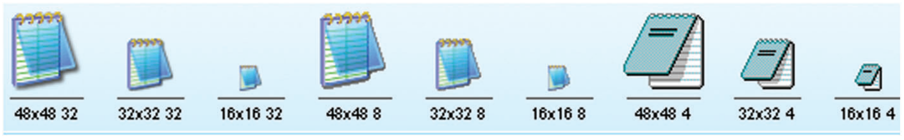

Figure 1: Icons of notepad program

The icon resource in the PE file is defined by two structures, RT_GROUP_ICON and RT_ICON. The icon group resource is a predefined resource type in the PE file, and its resource type ID number is 0EH, which is equivalent to the icon header and icon item in the ICO file. Each icon group resource describes the basic information of multiple icons with similar shapes, sizes or numbers of colors, mainly including the number of icons contained in the icon group, the width, height, color number, and icon ID number of each icon. Wait. If there are multiple icons in the PE file, there must be an icon group in the resource section. Icon RT_ICON is the icon data part, and each icon data corresponds to an icon resource. For all icon group resources in the PE file, traverse all the icons in each icon group in turn to achieve access to all icons.

2.2 Data Structure of Icon

In the resource section, Icon resources are commonly described by icon group resources (RT_GROUP_ICON) and icon resources (RT_ICON). The RT_GROUP_ICON resource is analogous to the ICONDIR data in the ICO file. One RT_GROUP_ICON resource is created for each ICO file bound to the EXE or DLL with the resource compiler/linker. The RT_GROUP_ICON resource is a simply GRPRESDIR structure. The RT_GROUP_ICON resource is composed of the GRPICONDIR structure and idcount GRPICONDIRENTRY structures. The idcount member indicates how many images are present in the icon resource. The GRPICONDIRENTRY structure describes the basic properties of the icon, providing details about its size and color depth. The nID field is the ID of the corresponding RT_ICON data resources, the corresponding icon data can be found by using ID number to traverse the directory tree of resources. Icon resource is composed of the bitmap header, palette, XOR bitmap and AND bitmap (more than 8-bit icon without palette). The XOR bitmap is the color portion of the image and is applied to the destination by using the XOR operation after the application of the AND mask, which describes the basic shape of the image. The AND bitmap is applied to the destination by using the AND operation, to preserve or remove destination pixels before applying the XOR mask, which determines the icon transparent area. The GRPICONDIR structure and GRPICONDIRENTRY structure are defined as:

```
typedef struct
{ WORD idReserved;       // Reserved (must be 0)
WORD idType;             // Resource type (1 for icons)
WORD idcount;            // How many images?
GRPICONDIRENTRY idEntries [1]; // The entries
} GRPICONDIR, *LPGRPICONDIR;
typedef struct
{ BYTE bWidth;           // Width, in pixels, of the image
BYTE bHeight;            // Height, in pixels, of the image
BYTE bColorCount;        // Number of colors in image
BYTE bReserved;          // Reserved
WORD wPlanes;            // Color Planes
WORD wBitCount;          // Bits per pixel
DWORD dwBytesInRes;      // bytes in this resource?
WORD nID;                // the ID
} GRPICONDIRENTRY, *LPGRPICONDIRENTRY;
```

3 Display Principle of PE File Icon

PE file icon is a transparent bitmap, only the irregular local bitmap images can be displayed in the Explorer, the other parts of the bitmap are transparent. In the transparent areas, the screen background becomes visible, which makes icons and background harmoniously blended. PE file icon consists of two separate bitmaps, called AND bitmap (or mask) and XOR bitmap (or image, or when we get to color icons, color). The AND bitmap is composed of white background and black icon images, which will be applied to the AND operation with the current screen. Hence, it is called AND bitmap. The XOR bitmap is composed of black background and color icon. The bitmap will be applied to the XOR operation with current screen, which is called the XOR bitmap. Drawing an icon takes place in two steps:

Step 1: temp bitmap = AND bitmap $\wedge$ screen background

Step 2: transparent bitmap = XOR bitmap $\oplus$ temp bitmap

where $\wedge$ denotes the AND operation, $\oplus$ denotes the XOR operation, transparent bitmap is composed of background image and color image.

In other words,

pixel = (screen $\wedge$ mask) $\oplus$ image

For the icon with transparent channel, the operating system invokes DrawIcon (DI_NORMAL) function to achieve alpha blending the image and the destination image in order to achieve a transparent icon display. One can see from the previous formula, for the pixel with the 0 Alpha channel value, the system will show the background color, the corresponding RGB values are useless, which can be used for information hiding. At the same time, the AND bitmap data of icon loss the meaning of transparent display (replaced by the Alpha channel value), so it can be used for information hiding.

4 Data Hiding Algorithm

For 16 color and 256 color icons, there is a palette data area, and each color is described by an RGBQUAD array element. As shown in the previous section, each RGBQUAD array element takes up 4 bytes, that is, for 16 color icons, the size of the palette is 64 bytes; for 256 color icons, the size of the palette is 1024 bytes [13–20]. For the 16 color and 256 color icons with color palette, the value of the icon XOR bitmap data area is the pixel color index value. When displaying, the RGB value of the pixel can be obtained by searching the palette data area with this index value. Experiments show that for 16 color and 256 color icons, only some color items are used, that is, redundant color items exist in the palette, see Tab. 1.

Table 1: 16-color and 256-color icon statistical characteristics

File name	Icon ID	Icon size	Palette colors	Unused colors
notepad.exe	4	16 × 16	16	9
	8	16 × 16	256	7
	3	24 × 24	16	7
	7	24 × 24	256	104
	2	32 × 32	16	5
	6	32 × 32	256	73
	1	48 × 48	16	5
	5	48 × 48	256	39
calc.exe	5	16 × 16	16	8
	10	16 × 16	256	187
	4	24 × 24	16	7
	9	24 × 24	256	149
	3	32 × 32	16	7
	8	32 × 32	256	104
	2	48 × 48	16	6
	7	48 × 48	256	51
	1	64 × 64	16	3
	6	64 × 64	256	50

According to the working principle of transparent display of icons, when the icons are displayed on the screen, only irregular icon graphics will be displayed, and the background color of the screen will be displayed in other parts. In this way, the working principle of transparent display with palette icons can be used for information hiding.

4.1 Secret Message Embedding

An icon is an irregularly shaped graphic. According to statistics, most icons have transparent pixels accounting for about 32% of the total pixels of the entire icon. When the windows operating system displays icons, these pixels will not be displayed. Instead, the screen background will be displayed.

However, since these pixels and the screen will perform XOR operation, they must be black (0). At the same time, for 16-color or 256-color palette icons, the palette usually has redundant color items. In this way, these unused color items can be used to count the unused color items, and these unused color items can be added to the palette. The RGB values of the color items are all changed to 0, that is, multiple color items whose components of RGB are all 0 (black) are repeatedly set, and then a black index value is selected in the XOR bitmap according to the rules, and these indexes are The value is encoded for information hiding. For each icon, you can create an index table and count its hidden capacity through the following methods. After the information is hidden, it will not affect the display of the icon, nor will it affect the execution results and performance of the program. Traverse all 16-color and 256-color palette icons in the PE file to get the total hidden capacity, see Tab. 2.

Table 2: Index coding table

Serial number	Index value	Coding
0	C0	0000
1	C1	0001
2	C2	0010
3	C3	0011
4	C4	0100
5	C5	0101
6	C6	0110
7	C7	0111
8	C8	1000
9	C9	1001
10	C10	1010
11	C11	1011
12	C12	1100
13	C13	1101
14	C14	1110
15	C15	1111

The detailed algorithm is as follows:

Input: secret information M, PE file F, key k

Output: the watermarked PE file F’

Step 1: encrypt the information M by using DES algorithm, produce cipher text M’. Calculate the byte length of the encrypted information, use two bytes to express the length and place it before the encrypted information, together with the encrypted information as the actual hidden information.

Step 2: Select the PE file, analyze the resource section of the PE file, and determine whether there are 4-bit and 8-bit palette icon resources, and calculate hiding capacity. If the hiding capacity is not enough, then select the other PE files;

Step 3: Count the hidden capacity of all 4-bit and 8-bit icon resources with palettes in the PE file. If the hidden capacity is less than the length of the actual hidden information, exit. The calculation process of hidden capacity is as follows:

a) Count the number of unused colors in the icon, set it to N, and record its index value, using C_1, C_2, $C_{i,}$ … C_N ($1 \le i \le N$).

b) In the palette, change the RGB values of the first $2^L - 1$ ($2^L - 1 \le N$) unused color items with the index value to 0, and get the color items with 2^L colors as black. Use C_0 respectively, C_1, C_2, C_i … ($0 \le I \le 2^L - 1$). At the same time, an index table is established for these color items and encoded with L binary bits.

c) Analyze the data of the AND bitmap and calculate the number of transparent pixels P;

d) $L \times P$ is the hidden capacity of the icon. Traverse all 4-bit and 8-bit icons with palettes in the PE file to get the total hidden capacity. If the hidden capacity is not enough, increase the size of resource section of the PE file, and copy one or more icons with palettes from the original icons to the resource section.

Step 4: The encrypted information is regarded as a series of binary bit streams, and these bit streams are divided into groups. The length L of the bits in each group is determined by the number of unused colors N in the icon, $L = \lfloor \log_2(N+1) \rfloor$.

Step 5: Sort the icons of 16 colors and 256 colors in ascending order according to the icon ID number.

Step 6: According to the icon ID number in ascending order, the encrypted information will be hidden in the corresponding icon in turn. The specific method is: analyze the value of the icon AND bitmap, scan the XOR bitmap in turn, if the value corresponding to AND is 1, it means that the pixel corresponding to the XOR bitmap will be displayed transparently, which can be used to hide information, according to the information group to be hidden Select an index value from the index table, otherwise skip the pixel.

4.2 Extracting Secret Information

The algorithm analyzes the watermarked PE file and extracts the secret information based on the principle of palette icon transparent display. The main steps are listed as following:

Input: the watermarked PE file F', key k.

Output: secret information M.

Step 1: read the secret PE file, record the 4-bit and 8-bit icons of the PE file with a palette, and sort them in ascending order of the icon ID number.

Step 2: Scan the palette of icon data in ascending order of icon ID number, count the color items of black (each component of RGB value is 0), set N kinds, take the first 2^L kinds (excess ones are not used), and record their Index value, build an index table based on these index values and encode.

Step 3: Analyze the value of the AND bitmap, scan the XOR bitmap in turn, if the value corresponding to the AND bitmap is 1, the pixel corresponding to the XOR bitmap hides information, look up the index table according to its index value, and extract the hidden length of L bits Binary information, otherwise skip the pixel.

Step 4: Assemble the information extracted in Step 3, and delete the extra bytes at the end according to the length value represented by the first two bytes to obtain the hidden information.

Step 5: Use the key to decrypt the extracted hidden information with DES to obtain the original information M.

5 Analysis of Experimental Results

Use this algorithm to conduct information hiding experiments on 16-color and 256-color icons contained in common software such as notepad.exe, calc.exe, etc. The experimental results are shown in Tab. 3.

Table 3: Analysis of icons with palette

File name	Icon ID	Icon size	Number of Colors	Number of unused colors	Hidden size per pixel (bit)	Number of trans-parent pixels	Total hidden capacity (byte)	Icon data size (byte)	Proportion
notepad. exe	4	16 × 16	16	9	3	43	16.125	128	0.1259
	3	24 × 24	16	7	3	213	79.875	288	0.2773
	7	24 × 24	256	104	6	213	159.75	576	0.2773
	2	32 × 32	16	5	2	410	102.5	512	0.2001
	6	32 × 32	256	73	6	410	307.5	1024	0.3002
	1	48 × 48	16	5	2	1024	256	1152	0.2222
	5	48 × 48	256	39	5	1024	640	2304	0.2777
calc.exe	10	16 × 16	256	187	7	65	56.875	256	0.2221
	9	24 × 24	256	149	7	145	126.875	576	0.2202
	8	32 × 32	256	104	6	338	253.5	1024	0.2475
	2	48 × 48	16	6	2	748	187	1152	0.1623
	7	48 × 48	256	51	5	748	467.5	2304	0.2029
	6	64 × 64	256	50	5	1450	906.25	4096	0.2212

Capacity: According to the embedding algorithm mentioned in this article, for each icon with a palette, its embedding capacity is related to two factors: one is the number of unused colors in the icon; the other is the number of transparent pixels in the icon. Assuming that the number of unused colors in the icon is N and the number of transparent pixels in the icon is P, the hidden capacity of each icon can be expressed as:

$$C = P \times \lfloor \log_2(N+1) \rfloor \tag{1}$$

Assuming that the number of 4-bit and 8-bit icons in the PE file is t, the total hidden capacity can be expressed as:

$$S = \sum_{i=1}^{t} (\mathrm{P} \times \lfloor \log_2(N+1) \rfloor) \tag{2}$$

Anti-attack: The algorithm utilizes the redundant data of palette icon to hide the secret information into the transparent areas of PE file icon resources and the AND bitmap area. The secret information is stored in its internal binary data as a part of PE file. As a carrier, the PE file and the hidden information are well blended. The PE file format is single, and its format and data cannot be transformed and modified easily, so it can effectively resist common format conversion attacks. The hidden information will not be destroyed applying the operation of compression, packed, unpacked to PE file. Subtracting, adding, distortion attacks all will cause PE file not to work in order. In addition, the hidden information is encrypted before embedding, which improves the performance of anti-attack.

Imperceptibility: In terms of concealment, on the one hand, this algorithm uses the feature of the palette that usually has redundant color items to embed information into the transparent area of the icon. After the information is embedded, the size of the icon does not change, that is, the size of the resource section will not change, and the size of the PE file will remain the same; on the other hand, the icon embedded in the data can be correctly parsed and displayed normally without any distortion. The PE file with the information embedded can run normally without affecting the performance of the program. So the algorithm has better concealment. Xu et al. [10] proposed an information hiding algorithm which embeds secret data into the palette of bitmap resources based on the characteristics of bitmap resources in PE files. The hidden information is too centralized and the security is poor.

6 Conclusion

It can be seen from the experimental results that the algorithm proposed has the following advantages: first, the algorithm makes use of the color redundancy of palette icons, and hides the information in the transparent area of the icon by encoding the index value, the display of the icon after the embedded information is exactly the same as the original icon, without causing any distortion; second, by analyzing the data of the AND bitmap, the information is written into transparent pixels that do not affect the display of the icon, the length of the PE file after the embedded information is no impact, and the concealment is good; third, the embedding and extraction algorithm only needs to analyze the palette icon, which is simple and easy to implement the algorithm complexity is low. and hiding capacity can be improved by increasing the number of call board icons. With the rapid development of Internet, it is popular for people to obtain software through the Internet, and at the same time, PE files are extensively used by computer users. Covert communication based on PE file has a good application prospects in the Internet environment. The algorithm utilizes the redundant data of palette icon to hide the secret information into the transparent areas of PE file icon resources and the AND bitmap area. The display of the icons embedded secret information is the same as the original icons. This method does not cause any visual distortion, without increasing the length of the program and affecting the performance of the program. The shortcomings of the presented algorithm: Modifying, adding or deleting palette icons will affect the extraction of information. The experiment shows that the algorithm has higher imperceptibility and higher hiding capacity than the existing PE file data hiding algorithm. The algorithm can be widely used for covert communication.

Acknowledgement: Thanks to the anonymous reviewers for their constructive suggestions to help improving this paper.

Funding Statement: This work is supported in part by National Natural Science Foundation of China (61373132, 61872408), the Key Laboratory of informationization technology for basic education in Hunan province (2015TP1017), Hunan provincial higher education reform research project (2012[528]), Project of innovation and entrepreneurship training program for college students in Hunan Province (2021[4045]).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] G. Duan, *Encryption and Decryption*. Beijing, China: Electronic Industry Press, pp. 405–448, 2018.

[2] J. P. Stern, G. Hachez, F. Koeune and J. J. Quisquater, "Robust object watermarking: Application to code," in *Proc. Information Hiding, Third Int. Workshop IH'1999*, Dresden, Germany, pp. 368–378, 1999.

[3] J. Q. Ai, X. M. Sun, Y. H. Liu, I. J. Cox and G. Sun, "A stern-based collusion-secure software watermarking algorithm and its implementation," in *Proc. 2007 Int. Conf. on Multimedia and Ubiquitous Engineering*, Seoul, Korea, pp. 813–818, 2007.

[4] Z. Q. Wu, S. D. Feng and J. F. Ma, "A scheme and implementation of information hiding based on PE file," *Computer Engineering and Applications*, vol. 41, no. 27, pp. 148–150, 2005.

[5] A. A. Z. Zaidan, B. B. Zaidan, A. W. Naji, F. Othman, A. A. Ahmed *et al.,* "Approved undetectable-antivirus steganography for multimedia information in PE-File," in *Proc. Computer Science & Information Technology-Spring Conf.*, Singapore, IEEE, pp. 437–441, 2009.

[6] H. Alanazi, H. A. Jalab, A. A. Zaidan and B. B. Zaidan, "New frame work of hidden data with in non multimedia file," *International Journal of Computer Science and Network Security*, vol. 2, no. 1, pp. 46–53, 2010.

[7] W. S. Fang, L. P. Shao and K. J. Zhang, "The research of information hiding technology based on portable executable file format," *Microcomputer Information*, vol. 22, no. 11, pp. 77–80, 2006.

[8] X. J. Xu, X. Y. Xu and H. H. Liang, "Information hiding scheme based on PE file resources section," *Journal of Computer Applications*, vol. 27, no. 3, pp. 621–623, 2007.

[9] Q. F. Duanmu, Y. B. Wang and K. Z. Zhang, "Information hiding scheme based on PE file resources datum," *Computer Engineering*, vol. 35, no. 13, pp. 128–130, 2009.

[10] J. Xu, J. F. Li, L. Y. Ya and Y. Wu, "An information hiding algorithm based on bitmap resource of portable executable file," *Journal of Electronic Science and Technology*, vol. 10, no. 2, pp. 181–184, 2012.

[11] Y. B. Luo, *32-bit Windows Assembly Language Programming*. Beijing, China: Electronic Industry Press, pp. 128–160, 2006.

[12] L. Qi, *Windows PE: The Definitive Guide*. Beijing, China: Machinery Industry Press, pp. 197–243, 2011.

[13] H. M. Liu, Z. F. Zhang and J. W. Huang, "A high capacity distortion-free data hiding algorithm for palette image," in *Proc. Int. Sym. Circuits and Systems, ISCAS'03*, Bangkok, Thailand, pp. 916–919, 2003.

[14] J. Guan, J. Li and Z. Jiang, "The design and implementation of a multidimensional and hierarchical web anomaly detection system," *Intelligent Automation & Soft Computing*, vol. 25, no. 1, pp. 131–141, 2019.

[15] Z. Zhou, J. H. Qin, X. Y. Xiang, Y. Tan and Q. Liu, "News text topic clustering optimized method based on TFIDF algorithm on spark," *Computers, Materials & Continua*, vol. 62, no. 1, pp. 217–231, 2020.

[16] Z. H. Xia, L. H. Lu, T. Qiu, H. J. Shim, X. Y. Chen *et al.,* "A privacy-preserving image retrieval based on AC-coefficients and color histograms in cloud environment," *Computers, Materials & Continua*, vol. 58, no. 1, pp. 27–43, 2019.

[17] L. Xiang, S. Yang, Y. Liu, Q. Li and C. Zhu, "Novel linguistic steganography based on character-level text generation," *Mathematics*, vol. 8, no. 9, pp. 1558, 2020.

[18] L. Xiang, G. Guo, Q. Li, C. Zhu, J. Chen *et al.,* "Spam detection in reviews using LSTM-based multi-entity temporal features," *Intelligent Automation & Soft Computing*, vol. 26, no. 6, pp. 1375–1390, 2020.

[19] Z. Yang, S. Zhang, Y. Hu, Z. Hu and Y. Huang, "VAE-Stega: Linguistic steganography based on variational auto-encoder," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 880–895, 2021.

[20] J. Qin, Y. Cao, X. Xiang, Y. Tan, L. Xiang *et al.,* "An encrypted image retrieval method based on SimHash in cloud computing," *Computers, Materials & Continua*, vol. 63, no. 1, pp. 389–399, 2020.