**TELKOMNIKA Indonesian Journal of Electrical Engineering**
Vol.12, No.2, February 2014, pp. 1012 ~ 1021
DOI: http://dx.doi.org/10.11591/telkomnika.v12i2.4215


# A Software Watermarking based on PE File with Tamper-proof Function

**Shengbing Che*, Yemao Wang**
School of Computer and Information Engineering, Central South University of Forestry & Technology, Changsha, Hunan, China
*Corresponding author, e-mail: cheshengbing727@163.com

***Abstract***

*After embedding the watermark based on the algorithm of PE file resources section, the data of resource section was still easy to be rearranged, so this paper put forward a new watermark algorithm with the function of anti-tamper. The new algorithm still used watermark information to control the order of software resource structure's nodes, and hid the watermark information in the rearranged resource structure. At the same time it recorded the exchanged node information of each layer, and encoded these exchanged nodes' information. Then it constructed corresponding MPPCT and DPPCT according to the encoding results to achieve the purpose of anti-tamper. Experiment shows that it realizes a blind detection, resists multiple common attacks effectively, and has strong robustness.*

***Keywords:*** *software watermark, PE file, resource section, anti-tamper*

***Copyright © 2014 Institute of Advanced Engineering and Science. All rights reserved.***

## 1. Introduction

The rapid development of internet makes the digital information spread conveniently, the digital products transmitted by internet, may be abused by vicious individuals or groups without the copyright license. So the copyright protection of the digital products is a problem urged to be solved, software watermark technology is proposed under this situation. It is a branch of digital watermark technology [1, 2], which is used to testify the software's copyright, also a vital branch of information hiding technology. The digital watermark technology means the operation of embedding information to a digital work, such as software, audio, video, pictures, text and so on, without being notified. The software watermark is about to embed the copyright and user's authentication information to software, and reach the goal of software protection.

Literature [3, 4] has made a summarized description. According to the ways of extraction, it can be divided into static software watermark and dynamic software watermark, the static one stores in program's data section or the code section that can be extracted without running the program, so it has the advantages of conveniently building and embedding, also easily extracting, but its robustness is undesirable, it likely being attacked by various ways, such as transfiguration, superaddition, and truncation etc. However, the dynamic one stores in program's executing state, that is to say, the watermark itself won't be embedded to the program, Instead, it is some special code like a specified data structures, such as graph, tree or link list, that embedded to program, using their metaphor of the topology attributes to represent the watermark information, and the watermark extraction is hidden under the status of program executing. As mentioned in literature [5], it has a better robustness. Now the recent researchers at home and abroad, are mostly focused on the dynamic one.

Literature [6] proposes an algorithm based on PE file resource section. The idea of the algorithm is using the watermark to control resource section's node storing order, hides the watermarking in rearrangement resource structure. Although it has good robustness, can immune to most known attacks, also its detection was completely blind, through experiment, we find that, after embedding the watermark, the sequences of the data in resource section has been changed, which will lead a tampering attack.

According to the deficiency of literature [6], this paper proposes an algorithm with tamper-proof function.

*Received July 25, 2013; Revised September 5, 2013; Accepted October 4, 2013*

## 2. Software Watermark Algorithm Analysis

### 2.1. A Dynamic Software Watermarking Algorithm based on PPCT Enumeration Coding

The existing main coding method of dynamic graph topology structures as follows: Pareto chart coding, base K chain table coding, PPCT enumeration coding, etc. The base K chain table has the largest embedded quantity, and then Pareto chart, but these two methods have the poor performance against the attack. PPCT (planted plane cubic tree) is a kind of good properties of topology [7], as shown in Figure 1. Viewed formally, it is a special kind of binary tree. The existing study shows that, PPCT can become a kind of a robust watermark carrier, and has the best performance against the attack compared to base K chain table, Pareto chart and the traditional dynamic graph topology structure. But on the contrary, PPCT has the low embedded quantity. According to this deficiency, now most of dynamic watermark algorithms based on graph topology structure, is mostly proposed by improving PPCT data structure, so as to improve the data rate and have higher robust performance. The structure characteristics of PPCT as follows:

Firstly, there is a root node called Origin, this node has two pointers. Its left pointer point to the tree's lowest right leaf node, and its right pointer point to binary tree's the root node.

Secondly, each node has its own two pointers, left pointer and right pointer. Nonleaf node, its left and right pointer point to its own children, the leaf node's right pointer point to itself. however its left pointer must meet the following rules: the left lower node of nonleaf node's right subtree point to its left subtree's right lower node, the most left leaf node of the entire tree point to the origin node.

Finally, the number of PPCT representation is $C(n)=\frac{1}{n}C_{2n-2}^{n-1}$, if the number of leaf node is $n$.

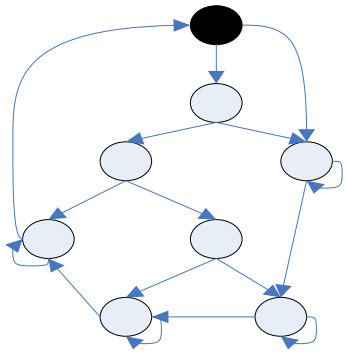

Figure 1. The Structure of PPCT

According to the deficiency of PPCT's low data rate, Literature [8] improves the right pointer of its leaf node, makes it contains certain information, we call this improved structure MPPCT. MPPCT structure retains the original structure's partial features, it still has an Origin node, namely the root node, and we just change the content of the leaf node's right pointer. The right pointer satisfies the following principles: for an integer, calculate its length of continuous 1 and 0 in binary data. modify the leaf node's right pointer, start from the most right leaf node, the leaf node's right pointer respectively point to the node which its serial number equal to the length of continuous 1 and 0.we express the binary data in this way. when we want to decode this structure, firstly we find out the right leaf node through the Origin node, decoding the number of continuous 1 which the leaf node represents in binary string, and so forth, when we decode the most left leaf node, the process of decoding is over, then we decode the binary string into decimal data form, which is the watermark information we want. In theory, MPPCT structure has $M/2$ leaf node if it contains $M$ nodes, the right pointer of each leaf node has $A_{M-1}^{1}$ nodes which it can point to, and so MPPCT structure can represent $C(M)=\left(A_{M-1}^{1}\right)^{M/2}=(M-1)^{M/2}$ data. Compared with PPCT, MPPCT data rate dramatically improved, also, its robustness has not affected.

At the same time, literature [9] puts forward a dynamic self-checking software watermark tamper-proofing method, which is specific language unrelated (we call it DPPCT), it expands watermark structure based on the dynamic graph structure, makes each node has its own integer ID field, and assign a random initial value to it. Then the structure is divided into sub-blocks and calculated by linear hashing function, this method includes both the sequence information and the quantity information of the nodes, also, a constant migration technique wraps the integrity checking in the normal logical judgment operation of the watermarked program. Due to that the authentication code of the watermark structure is self-contained, tampering the structure will lead to the application's undefined behavior, Analysis results show that this algorithm is capable to withstand multiform attacks and its protective capability is strong.

### 2.2. Watermark Algorithm based on PE File Resource Section Format

At present, win32 execution body under Windows systems are all based on PE file format. Its wide application makes the software watermark based on this file format imminent. Currently, this kind of algorithm mainly includes three categories. The first as mentioned in literature [10], this kind of method using the redundancy space of PE file to embed the watermark information, so it's easy to perceive, and has poor robustness. The second as presented in literature [11], watermark is constructed by using the SMC technology and the length of instructions to encode watermark. Also, there is an algorithm that based on the instruction encoding of PE file, this kind of method is a native code implement of technical solutions by analyzing to find the replacement instructions and encode it, read the sequence they appear and change the sequence to achieve the watermark information extraction and embedding. This kind of method is easy to conceal, but hard resisting against confusion. The third as mentioned in literature [6, 12, 13] which is based on structural characteristics of PE file format, such as resource section and import table, this kind of method don't add or modify the redundant space and instruction of PE file, so it has higher imperceptibility than the first and the second type. Now, as mentioned in literature [14], in the process of research, many researchers also introduce the reverse engineering technology and chaotic systems to improve the robustness of the watermark.

As we know, there're lots of resources in an executable file based on PE file format, such as ICONS, BITMAP, DIALOG BOX, SOUNDS and so on. They store in resource section like a similar tree data structure. As shown in Figure 2.

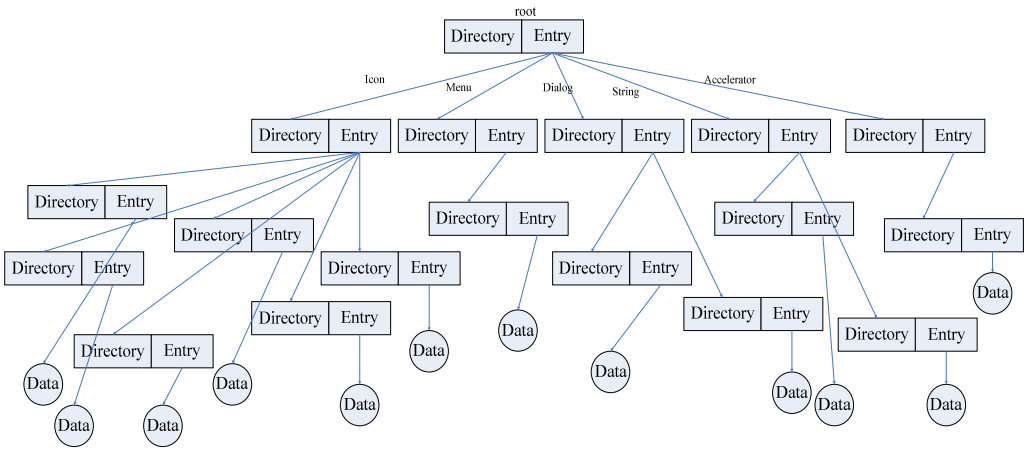


Figure 2. The Structure of PE Resource Section

Resource section usually consists of 3 layers, its top is resource type, and then name, the last is the language. At the start of the resource section is a data structure of IMAGE_RESOURCE_DIRECTORY which is followed by the array of IMAGE_RESOURCE_DIRECTORY_ENTRY, each element of the array represents different resource types; through each element, we can find the second layer's IMAGE_RESOURCE_DIRECTORY, also followed

by the array of IMAGE_RESOURCE_DIRECTORY_ENTRY. This layer, its array's each element represents different resource name; then, we can find the third layer's IMAGE_RESOURCE_DIRECTORY, also followed by the array of IMAGE_RESOURCE_DIRECTORY_ENTRY. This layer its array's each element represents different resource language; finally, through each IMAGE_RESOURCE_DIRECTORY_ENTRY, we can find the real resource. In figure3, its specific resources data information doesn't mark out, which should be placed in the fifth layer, it can be found by the fourth layer named Data. The process of finding a resource as follows:

Step 1: Read out resource table's PointerToRawData from PE file's section table.

Step 2: According to PointerToRawData find resource section's the starting position, and find the position of the next layer resource information through the first three layers' OffsetToDirectory in turn.

Step 3: Read out the information of OffsetToData from IMAGE_RESOURCE_DATA_ ENTRY in fourth layer, and then we can find the corresponding resource in fifth layer.

Literature [6] is just using resource section's special data structure to embed the watermark information. It uses the watermark to control resource section's node storing order, hides the watermark in rearrangement resource structure. Also, this algorithm introduces watermark authentication center (WAC), produces watermark information using technology of large-number-difficult-decomposition.

## 3. A Watermark Algorithm with Tamper-proof Function

### 3.1. Watermark Embedding

In order to make the resources structure express more clearly, we describe the resource section as five layer according to fig3.we use a long rectangle which followed by a group of short rectangle to represent a node in fig3, the long rectangle express the structural body of IMAGE_RESOURCE_DIRECTORY, the short rectangle express the structural body of IMAGE_RESOURCE_DIRECTORY_RNTRY. OffsetToDirectory which is the data member of IMAGE_RESOURCE_DIRECTORY_ENTRY, stores the relative virtual address (RVA) of the next layer's resource information, that is to say, it points to the next layer's node. The circle expresses IMAGE_RESOURCE_DATA_ENTRY in fourth layer. The diamond expresses the specific resource data that we can find it through OffsetToData, which is the data member of the IMAGE_RESOURCE_DATA_ENTRY, as shown in Figure 3.

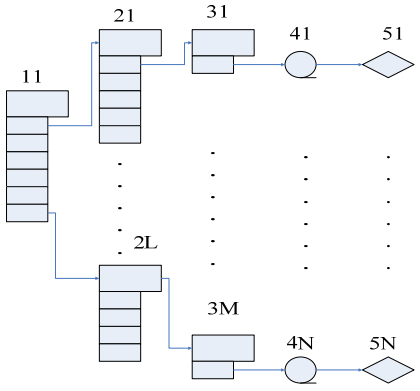


Figure 3. The Abstract Structure of PE Resource Section

Resources section usually use ID to identify resource types, its sequence is ranked in the order of smallest to largest. now, we mark each node in every layer, since there is only one node in layer 1, we mark its serial number as 11, then mark the second layer serial number as 21、22……2L according to the order of node's IMAGE_RESOURCE_DIRECTORY_ENTRY structural body in the first layer, and so forth, mark the third layer serial number as 31、32……3M, by analogy, we know that the fourth layer have a corresponding relationship with

the third one, also, the fifth layer have a corresponding relationship with the fourth one, so, mark the fourth serial number as 41、42……4M, mark the fifth serial number as 51、52……5M.

According to the experiment, if the same layer's nodes' sequence out of order in every layer, it is not sensitive for software to call resource information. When software is running, the resource information of each layer, its correct search is decided by the position value of OffsetToDirectory (or OffsetToData) in corresponding upper layer. if rearrange the order of the node in each layer, at the same time, modify the corresponding position value in upper layer, the resource information can still be called correctly, the software still has the original function, and can be executed normally.

The watermark algorithm based on resource section, controls the node storing order of software resources section structure according to the watermark, and hides watermark in rearrangement resources structure. The rule of node arrangement as shown in Figure 3, the first layer only has one node, so it cannot be used for arrangement, we just rearrange 2,3,4,5 layers' node. because the resource section is a very large data section, we just exchange the first node with another node in the same layer, according to this method, suppose the second layer has L nodes, the third one has M nodes, the fourth and the fifth also have M nodes, so we can get $L \times M \times M \times M$ kinds of arrangement scheme, each node's rearrangement correspond to an integer $Z$, we regard the integer $Z$ as watermark information. When embedding watermark information, firstly, we should rearrange the nodes in fifth layer; also, modify the corresponding address information in fourth layer, which point to the fifth layer's exchange node. By the same token, rearrange the node in fourth, third, second layer in turn, and modify the corresponding address information in the former layer, only in this way can the software call the resources correctly.

After embedding the watermark, we find that it's easy to rearrange the order of the node in every layer, in order to avoid the attacker changes the order of the node; we should take a tamper -proof mechanism.

### 3.2. A Tamper-proofing Mechanism

The purpose of tamper-proof mechanism is to protect the watermark's integrity, as mentioned in literature [15-17], once watermark is manipulated, the software can be immediate percept, and then terminate its operation. Literature [8] puts forward a tamper-proof software watermark using code-based encryption, changes to the source and object code are made to embed the watermark, and according to certain policies, some parts of the object code is encrypted with an en/decryption key that is highly coupled with the object code to increase robustness and tamper-proof capability. Literature [18] proposes a dynamic graph software watermarking scheme based on variable tamper-proofing, it encodes the watermark with a parent pointer of the dynamic graph node, and combines graph node and the original program variables for tamper-proofing. The result of experiment shows that the watermark is not obviously effective in changing the program speed and results, and can resist many types of attacks.

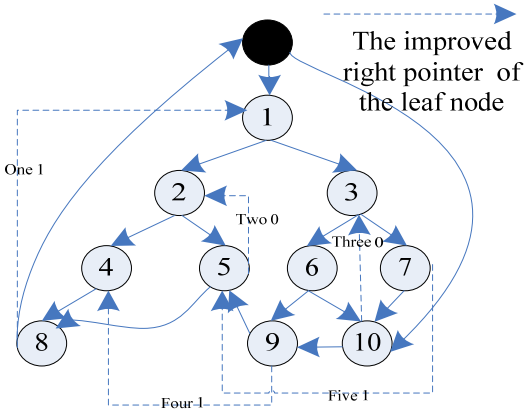


Figure 4. The Structure of MPPCT

We use the thought of literature [8, 9] to realize a tamper-proof mechanism. At the beginning, we should know the method of data encoding, its specific process is as follows:

Step 1: Record the exchanged node's index information in each layer, mark them $m_1, m_2, m_3, m_4$, then, turn these decimal data to binary string, mark them $(m_1)_b, (m_2)_b, (m_3)_b, (m_4)_b$, then, calculate the length of continuous 1 and 0 in binary data, and mark them $l_1, l_2, \cdots, l_n$ (n is the number of leaf nodes).

For example, suppose that, the exchanged node's index in the second layer is 15, the third one is 8, the fourth one is 7, the fifth one is 9, these decimal data are 1111, 1000, 111, 1001, so, we get the encoding result is 1111 1000 111 1001.

Step 2: Use the data structure of MPPCT to express the encoding result. Among them, n=5, $m_1$=15, $m_2$=8, $m_3$=7, $m_4$=9, $l_1$=5, $l_2$=3, $l_3$=4, $l_4$=2, $l_5$=1, As shown in Figure 4.

After getting the MPPCT, we can use the method mentioned in literature [19, 20] to embed it into the PE file. Then, we should realize the tamper-proof function, its basic principle and concrete implementation process is as follows:

Step 1: Extend the structure of MPPCT, make its each basic element contain an integer ID field, and then assign a random initial value for each ID field. As shown in Figure 5, each value represents the corresponding random number.

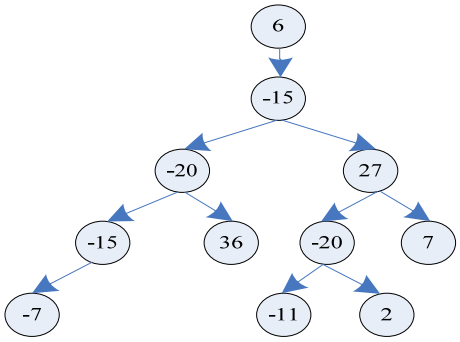


Figure 5. MPPCT with Inserted ID Field

Step 2: MPPCT is divided into many sub structure, at least one intersection element between adjacent sub graphs, as shown in Figure 6. The original MPPCT is divided into 3 sub graphs, among them, the dark nodes are the intersection between graphs.

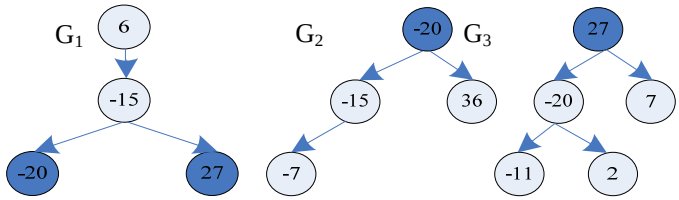


Figure 6. Three Sub-graphs

Step 3: Generate an integrity verification code for each sub graph. Its purpose is to trigger the appropriate events when the watermark is attacked. So, each sub graph is calculated by the linear hashing function, and the hash values should be a constant.

The rule of calculating the hash value is as follows:

Assuming that, the total number of MPPCT's nodes is N, the number of leaf nodes is n, we number MPPCT's nodes according to the preorder traversal method (starting at 1), and

make the product between the serial number of each node and its ID field as $d_i$, then the corresponding hash value $h_i(G_j)$ can be recursively defined as follows:

$$h_0(G_j) = C \quad (1)$$

$$h_i(G_j) = c \times (d_i + h_{i-1}(G_j)) \quad (2)$$

Among them, $0 < i \le N$, $C = \sum_{t=1}^{4} m_t - m_{j\%5}$, $c = (\sum_{t=1}^{n} l_t - l_{j\%(n+1)})\%5 + 1$.

In order to guarantee the hash value is equal to a given constant, we can adjust the sub graph's last traversal element's ID field, if a revised sub graph its last node's ID field is not an integer, we should adjust two or more than two nodes to make sure that the hash value is a constant and the whole ID fields are an integer.

According to the known conditions above, we know that $h_0(G_1)$=24, c=1. Below, we use $G_1$ to show the process of node's modification, as shown in Table 1.

Table 1. Hash Value Calculation of Subgraph G1

| Node | ID value | $d_i$ | Hash value | The new ID value |
|---|---|---|---|---|
| 1 | 6 | 6*1 | 30 | |
| 2 | -15 | -15*2 | 0 | |
| 3 | -20 | -20*3 | -60 | |
| 4 | 27 | 27*4 | 48 | 60 |

In order to enhance the protection of the watermark, we only calculate the first two sub graph's hash value, that is to say, we need not calculate $G_3$'s individual verification code, the method to calculate $G_3$'s verification code is to combine itself to $G_1$ and $G_2$, then calculate the combined graph's hash value. As shown in Figure 7.

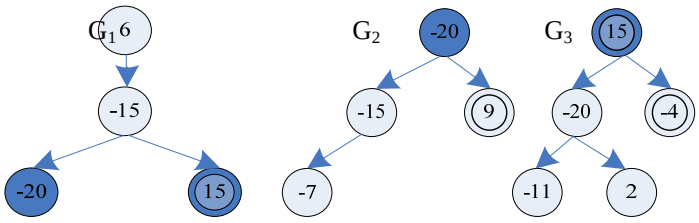


Figure 7. After Revising the Checksum

Step 4: Insert the verification code into the program's appropriate placement. Because the entire subgraph's hash value are a constant, that is to say, they can be easily mapped to Boolean true or false, on the other hand, there are large amounts of conditional statements in a program, these hash values can be easily hidden in it. If you want to achieve stronger protective effect, we should revise the hash value, make the value equal to the corresponding constant value, and then it can be arbitrary replacement in the application of constants.

### 3.3. Watermark Extraction

First of all, we should ensure that the resource section isn't tampered, then, start to extract the watermarking information, its extraction method is as follows:

Step 1: Calculate the number of each layer's node, suppose that, the second layer has $d_1$ nodes, the third is $d_2$, the fourth is $d_3$, and the fifth is $d_4$.

Step 2: The index of exchanged node in each layer, they are $mark_1, mark_2, mark_3, mark_4$ (index starts from 0).

Step 3: The computation formula of watermark information is as follows:

$$mark_1 \times (d_2 \times d_3 \times d_4) + mark_2 \times (d_3 \times d_4) + mark_3 \times d_4 + mark_4. \quad (3)$$

## 4. Experimental Result and Algorithm Performance Analyses

### 4.1. Experimental Result

This experiment regards the WRITE.EXE as the carrier of software. As shown in Figure 8, this figure illustrates the structure of WRITE.EXE's resource section before and after embedding the watermark, it proves that this algorithm has strong imperceptibility. Figure 9 illustrates the data information of the first node and the fourth node in the fourth layer; they are stored in the form of IMAGE_RESOURCE_DATA_ENTRY. The red rectangle marks the address information in the third layer, which respectively point to the exchanged nodes in fourth layer, these information are stored in OffsetToData, which is the data member of IMAGE_RESOURCE_DIRECTORY_ENTRY. The first node's information in fourth layer is marked by the green wave line; the fourth node's information is marked by the blue wave line. After embedding the watermark information, as shown in Figure 10, not only the information between the first node and the fourth node is exchanged, but also the information in the third layer which point to the exchanged nodes information in fourth layer, is also exchanged. While the part of tamper-proof mechanism is the record of the exchanged node information in resource section's each layer, the sequence of the nodes in the resource section would change in the case that the watermark information is distorted. Once start the tamper-proof mechanism, the software will fail to execute, which achieved the goal of avoiding distorting, and kept the watermark information integrity.

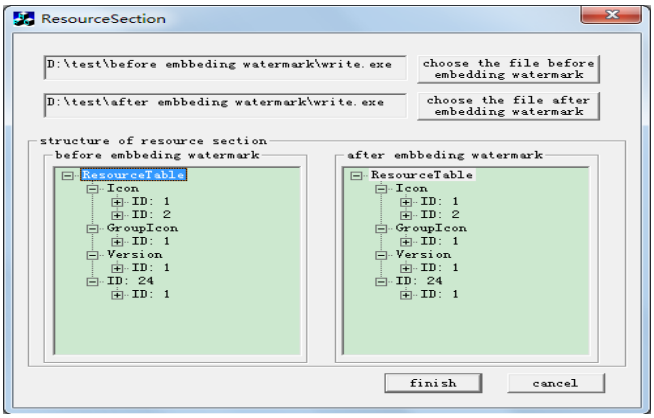


Figure 8. The Structure of Resource Section before and after Embedding the Watermark

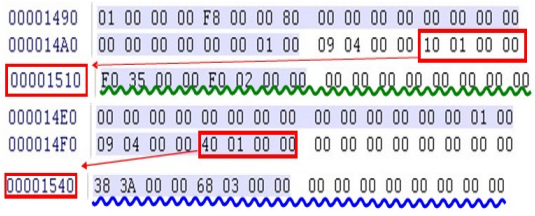


Figure 9. Before Embedding the Watermark

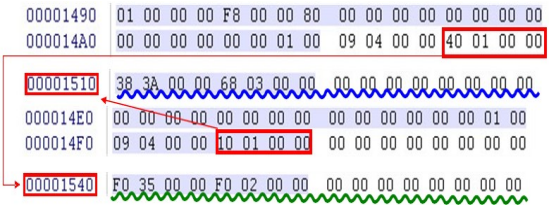


Figure 10. After Embedding the Watermark

### 4.2. Algorithm Performance Analyses

According to the experiment, this algorithm is very ideal transparency. First of all, it doesn't use the PE file's redundancy to embed watermark, on the contrary, uses the PE file's structure features to embed watermark, makes the hidden information and the file well mix together. The process of extracting watermark also doesn't need original PE file, fully realizes a blind detection. Also introduces a tamper-proof mechanism, uses PPCT data structure's characteristic, and combines the MPPCT and DPPCT to realize the watermark information prevent manipulation functions, it enhance the algorithm's robustness. First of all, it can successfully resist static analysis, in theory, the attacker can find all of the validation code through static analysis and the corresponding hash function, but the hash function is treated with bewilderment, it would be very difficult to analyze; Secondly it can resist dynamic analysis, if the attacker wants to dynamic track the code by running the program, it must analyze all jump statements, it will require a lot of time and energy for large software; At the same time, to a certain extent, it also can resist pattern matching attack, it will observe program execution by some tools such as the debugger and so on, and monitor the call number of some function, then try to track them. Due to the hash function has several different versions, and treated with the confusion, it is hard to be observed, and this attack will not get more information than other attack methods.

## 5. Conclusion

Software watermarking is a cross research field that contains cryptography, software engineering, algorithm design, graph theory, program design disciplines and so on. It solves the problem of software copyright protection. At present, PE file format is widely used in windows system, this paper analyses its characteristics of resource section structure in detail, then proposes a tamper-proof mechanism based on MPPCT and DPPCT data structure. The new algorithm fully realizes a blind detection, also can effectively resist a variety of common attack and improve the robustness of the watermarking. But the completion of its tamper-proof function still needs some manual work, also, the software exist behavior deterministic, execution environment dependence etc, it makes attackers still have sufficient information to understand the software behavior characteristics, so, how to protect the integrity of the watermark information is still an important issue that need further study. At the same time, the study is mainly focused on the single watermark algorithm, which is difficult to resist various attacks. So, we should pay more attention to the study of multi-watermark, With further study, the protect technology based on multi-watermark need more work, includes automatization of software multi-watermark loading, remote authentication based on watermark, how to balance the elusiveness, data rate and robustness of the software watermark, which provides a new research direction for the later researcher.

## Acknowledgement

This paper was supported by the Graduate's Scientific Research Foundation of Central South University of Forestry and Technology (Grant No.CX2012B09).

## References

[1] Qiming Liu. An adaptive blind watermarking algorithm for color image. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(1): 302-309.
[2] Huming Gao, Liyuan Jia, Meiling Liu. A digital watermarking algorithm for color image based on DWT. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(6): 3271-3278.
[3] Yang Jianlong, Wang Jianmin, Li Deyi. Recent Development of Software Watermark. *Computer Engineering*. 2007; 33(17): 168-175.
[4] Zhang Lihe, Yang Yixian, Niu Xinxin, Niu Shaozhang. A Survey on Software Watermarking. *Journal of Software*. 2003; 14(2): 268-277.
[5] Collberg C, Thomborson C, Townsend G. Dynamic graph-based software fingerprinting. *TOPLAS*. 2007; 29(6): 1-67.
[6] Xu Xiaojing, Xu xiangyang, Liang Haihua. Watermark algorithm based on PE File Resource Section Format. *Computer Engineering and Design*. 2007; 28(23): 5802-5804.

[7] Zhu Jianqi, Liu Yanheng, Yin Kexin. A Novel Planar IPPCT Tree Structure and Characteristics Analysis. *Journal of Software*. 2010; 5(3): 344-351.
[8] Tang Zhanyong, Fang Dingyi, Su Lin. A tamper-proof Software Watermarking Using Code-based encryption. *Journal of University of Science and Technology of China*. 2011; 41(7): 99-606.
[9] Zhang Xuesong, Li Xin, Cui Weili, He Fengling, Zuo Wanli. Dynamic self-checking software watermark tamper-proofing method. *Journal of Jilin University (Engineering and Technology Edition)*. 2007; 37(5): 1148-1153.
[10] Liu Hongsheng, Sun Changying. The Protection of Software Copyright by Software Watermarking. *Computer Knowledge and Technology*. 2010; 6(7): 1701-1703.
[11] Zhou Ping, Yang Xu Guang. PE Software Watermarking Encoded based on Length of Instructions. *Computer Engineering and Design*. 2008; 29(19): 5119-5122.
[12] Zuwei Tian, Xingming Sun, Hengfu Yang. Information Hiding Algorithm based on PE File Icon Resource. *JCIT*. 2011; 6(6): 469-475.
[13] Long Feiyu, Liu Jiayong, Yuan Xi. Software watermark based on structure transform of PE file import table. *Journal of Computer Applications*. 2010; 30(1): 217-219.
[14] Lu Bin, Luo Xiangyang, Liu Fenlin. A Chaos-Based Framework and Implementation for Software Watermarking Algorithm. *Journal of Software*. 2007; 18(2): 351-360.
[15] Collberg C, Thomborson C. Watermarking, tamper-proofing and obfuscation-Tools for software protection. *IEEE Transactions on Software Engineering*. 2002; 28(8): 735-746.
[16] Jianqi Zhu, Yanheng Liu, Aimin Wang. H Function based Tamper-proofing software Watermarking Scheme. *Journal of Software*. 2011; 6(1): 148-155.
[17] Zhu Jianqi, Liu Yanheng, Wang Aimin, Yin Kexin. H Function based Tamper-proofing Software Watermarking Scheme. *Journal of Software*. 2011; 6(1): 148-155.
[18] Li Shuzhi, Liu Meng. A Dynamic Graph Software Watermarking Scheme Based on Variable Tamper-Proofing. *Computer Engineering & Science*. 2011; 33(5):18-21.
[19] Wu Zhengqiang, Feng Shaodong, Ma Jianfeng. A Scheme and Implementation of Information Hiding Based on PE file. *Computer Engineering and Applications*. 2005; 41(27): 148-150.
[20] Hu Shan. Study on Inserting Executable Codes into PE Files. *Journal of Anshan University of Science and Technology*. 2005; 28(2): 119-122.