A Dynamic Graph Watermark scheme of Tamper Resistance

XiaoJiang Chen, DingYi Fang, JingBo Shen, Feng Chen, WenBo Wang, Lu He
College of Information Science and technology
Northwest University,
Xi’an, China
e-mail: dyf@nwu.edu.cn

Abstract - A novel approach of dynamic graphic software watermark is proposed. In this scheme, many fake watermarks created through encoding multi-constant, which’s structure is similar to the only true one’s, are introduced to enhance the stealth and anti-attack ability of dynamic graph watermark and keep software from sabotage. Meanwhile, a detailed analysis in theory is made in terms of the principle, feasibility and merits of this approach. A series of tests are made on the basis of the prototype system, analyzing the two sub-systems separately about their validity, robustness and performance overload caused by watermark embedding. Moreover, bit-rate of the three graph watermark structures in the system is analyzed in theory.

Key words: Software Watermark, Dynamic Graph Watermark, Tamper Resistance, Fake Watermark.

I. INTRODUCTION

With Dynamic Graph Watermark (DGW) [1,2] technique, watermark created dynamically can be transformed into some kind of graph topology hiding in software codes. The DGW use pointer or address reference to generate topological diagram. Each time running the program, we can map the logical addresses to different physical addresses so that the watermark can be hidden in the changing topological diagram. DGW can greatly enhance the robustness and imperceptibility comparing with the static watermark. At the same time, when the watermark are under attack, we can speculate the attacking means and even restore the original watermark according to some notable features changes of topology [3].

Graph topology selection, namely selecting the representation mode of watermark number, has become the key of DGW’s success [3,4]. At present, there are mainly two watermark topology in DGW: Radix-K Circular Linklist and Planted Plane Cubic Tree (PPCT). The former is easy to implement and full use of each pointer domain of graph node, but the watermark number construct is too simply. The latter can make more representation constructs for the same watermark number so that it can easily generate watermark bank, but the right pointer of the leaf node in the topology cannot be used to the full. The safety software system lab of United States Purdue University have done some experiment based on the generation method of PPCT watermark. However, they found that the results are not good, and the main reason is the shortage of good watermark representation method.

A kind of improved Planted Plane Cubic Tree (IPPCT) is proposed and used in our watermark system, which are the combination of the advantages of both the Radix-K Circular Linklist and the PPCT. Theory and Practice has proved that, compared with the former two IPPCT have better performance. On the basis of IPPCT, we can build a practical software watermark bank to resist the collusion attacks.

II. RADIX-K CIRCULAR LINKED LIST WATERMARK STRUCTURE

This watermark structure is a double pointer circular linked list (Figure 1-1).

In the structure, each node has two pointers, where the right pointer point to the next node and the left pointer is used to encode watermark information; The encoding information value is defined as the length of the path from the node back to itself, for example, a null-pointer represents 0, a self-pointer represents 1, a pointer to the next node represents 2, etc. In order to locate the watermark when extracting, we add an origin node before the first node of listlink. The origin node’s right pointer refers to the first node, and the left pointer is null.

A decimal watermark number can be decomposed in terms of radix-k:

\[ N = \sum_{i=0}^{k-2} e_k \times k^i \]

where \( 0 \leq e_i < k \). For example: the watermark number \( N = 4453 \), the factor \( R_1 = 61 \), \( R_2 = 73 \); \( N \) can be decomposed in the format of radix-6:

\[ N = 4453_{10} = 3 \times 6^3 + 2 \times 6^2 + 3 \times 6^1 + 4 \times 6^0 \]

Because of \( N \) = \( \sum_{i=0}^{k-2} e_k \times k^i \) and \( 0 \leq e_i < k \), one link structure including \( M \) nodes can represent any watermark number in the range 0 to \( M^{M-1} \), and the number of information bits that it can represent is \( \log_2 M^{M-1} \). In addition, every node has two pointers, so the radix-k list requires at least 2M words. According to the definition of the watermark structure bit-rate, the radix-k list’s bit-rate is: the radix-k list’s bit-rate is: 

\[ \log_2 M^{M-1} / 2M = \log_2 (M - 1) / 2 \]
III. PPCT WATERMARK STRUCTURE

The watermark structure based on PPCT evolved from the PPCT tree, which is a special binary tree (Figure 2-1(i)) formally. It can be slightly improved to be the PPCT watermark structure representing the DGW (Figure 2-1(ii)). It has following properties:

1. Every node has left and right pointer. No-leaf node’s two pointers point to its two subtrees respectively. The leaf node’s right pointer point to itself and its left pointer is ruled by: left-most child of no-leaf node’s right subtree is linked to the right-most child of its left subtree. The left pointer of the left-most child of the PPCT tree points to the Origin node.

2. The structure is enumerable. According to Catalan theory, the PPCT tree with m leaf node (all 2^{m} nodes) has C(m) = \frac{1}{m+1}C_{2m-2} type structures, where C(m) is Catalan number. The varieties number of the PPCT tree with 1-16 leaf nodes are: 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845. For example, Figure 2-2 shows the enumeration of the PPCT with 1-4 leaf nodes.

Then the definition is as following:

\[
\text{int}(T) = 0, \text{if } |T| = 1
\]

\[
\text{int}(T) = \text{int}(T.left) \times C(R) + \text{int}(T.right) + \text{min}_\text{int}(L, R), \text{if } |T| > 1, \text{min}_\text{int}(L, R) \text{ is the minimum integral number the T can represent, the recursive definition is as following:}
\]

\[
\text{min}_\text{int}(1, R) = 0
\]

\[
\text{min}_\text{int}(L, R) = \text{min}_\text{int}(L-1, R + 1) + C(L - 1) \times C(R + 1)
\]

\[
= \sum_{i=1}^{L-1} (C(L-i) \times C(R+i))
\]

Because of the double structure features of PPCT, namely the binary tree and linked list, we can use the pointer to construct the watermark. According to the character of the memory management, the pointer’s value is different when running, which will give the attacker a greater disturbance.

Meantime, the Origin node can be located within m-1 steps along the left pointer from any one node in the PPCT tree with m leafs so that traveling the whole tree can be implemented easily, which can greatly help to locate the watermark structure in the stack memory. Besides, once a node’s pointer is tampered, we can effectively restore its value according to the rule.

IV. THE IMPROVED PPCT WATERMARK STRUCTURE

The figure 3-1(i) shows a PPCT structure with four leaf nodes. All the nodes’ right pointer points to themselves, so we can use the right pointer of each node to contain certain extra information. Similar to the radix-k circular linked list structure, we can use the right pointer to encode watermark structure.

In the radix-k circular linked list structure, the watermark number N can be represented both by K-1 nodes (except for the origin node) in terms of radix-k: \( N = \sum_{i=0}^{k-1} e_i k^i, 0 \leq e_i < k \), and by IPPCT structure with m leafs in the terms of radix-m+1: \( N = \sum_{i=0}^{k-2} e_i k^i, 0 \leq e_i < k \). Coefficient e, can be obtained through the information contained in the corresponding right pointer which is just the number of leaf nodes that is traversed from current node to itself along the right pointer, namely, a null-pointer represents 0, a self-pointer represents 1, a pointer to the next node represents 2, etc. Figure 3-1(ii) shows a IPPCT structure with four leaves, from which the
watermark number \( N = 603 = 3 \times 5^3 + 0 \times 5^2 + 4 \times 5^1 + 4 \times 5^0 \), and
\[ N = \sum_{i=0}^{m-1} e_i \times (m+1)^i, \quad 0 \leq e_i < m+1, \text{ we can conclude that} \]
\[ 0 \leq N \leq (m+1)^m - 1 \] (m is the number of leaves). The max watermark number represented by the PPCT structure and the ICCPT structure with m leaf nodes are showed in table 3-1.

Table 3-1 The Max number of PPCT and IPPCT

<table>
<thead>
<tr>
<th>Num of leafnodes ( m )</th>
<th>Max num of PPCT structure</th>
<th>Max num of IPPCT structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 1.0 \times 10^6 )</td>
<td>( 1.0 \times 10^6 )</td>
</tr>
<tr>
<td>10</td>
<td>( 5.0 \times 10^3 )</td>
<td>( 2.6 \times 10^{10} )</td>
</tr>
<tr>
<td>100</td>
<td>( 2.0 \times 10^{56} )</td>
<td>( 2.7 \times 10^{100} )</td>
</tr>
</tbody>
</table>

For a DGW system, if the IPPCT method is adopted to represent 512 bits watermark number, compared with the PPCT structure, we only need 81 leaf nodes, reducing the time and space complexity for establishing watermark. Because the IPPCT still hold part features of PPCT, for m leaf nodes, we can choose any one \( \frac{1}{m} \cdot \sum_{i=0}^{m-1} e_i \) IPPCT structures to form a enough large watermark library to resist collusion attack.

V. FAKE WATERMARK ALGORITHM BASED ON MULTIPLE CONSTANTS ENCODING

A method of fake watermark with multiple constant encoding is proposed to defend attack to DGW, and a special DGW is created on the basis of method. The basic idea is that optimal constant is chosen and converted to the topology structure which is similar to the structure of Watermark graph, thus the many so called fake watermarks that can lead to the obfuscation for the attacker are formed. Attacking to any fake watermark will cause a break of program, except that attackers find the unique true watermark structure from many fake watermark structures and attack it. The method is as shown in the figure 4-1.

A. The selection of coding constant

The first, the program of watermark is lexically analyzed, and the optimal encoding constants is chosen and kept in user’s self-defined encoding constants list. For the file of Java class, the selection of encoding constants to the Watermark program is achieved by accessing the constant pool.

According to the watermark solutions with the difficulty of factoring large integer, the number transformed into graphic structure should be an integer, and so these non-integral constants in encoding constants list should be changed into integer.

B. Constant coding and constant decoding

Figure 4-2 True markwater compared with Fake markwater

The Constant coding is that the chosen optimal encoding constants \( C_1, C_2, \ldots, C_k \) (K is the number of chosen constant) are converted to graph topology, namely, creating the corresponding fake watermark graph \( CG_1, CG_2, \ldots, CG_k \). We will separately construct the encoding function \( \text{enCode}(\text{int }i) \) to complete the conversion of constants to graph, which is similar to the watermark dynamic generation in the DGW system. From the figure 4-2, the similitude can be seen clearly when the fake watermark graph is compared with the true watermark graph. Consequently, it can make the obfuscation for attackers so that they cannot nearly find the true watermark and attack it.

The constant decoding is the converse process of constant coding, which means to convert the fake watermark graph to the constants. For IPPCT structure, the number of leafs m is known and the coefficient \( e_i \) can be calculated as follows:

With the leaf node indexed by the left pointer of origin as starting node, \( L_1, L_2, \ldots, L_m \) represent the m different leaf nodes and m also is defined as the power of radix \( r \) then:

\[
C_i = \begin{cases} 
0 & j \leq i < j + 1, \quad i \leq j \\
\frac{1}{m-|j-i+1|} & j > i 
\end{cases}
\]

\( j \) is the subscript of the \( L_j \) indexed by the right pointer of \( L_i \).

Therefore, according to \( N = \sum_{i=0}^{m-1} e_i \times (m+1)^i \), the decoding function \( \text{deCode}(\text{IPPCT } CG[j]) \) can be constructed to complete the reverting process of the IPPCT structure to constant.
C. Capability analysis

In the DWG generated by fake watermark with multi-constants, the number of watermarks are represented by:

\[ N = \sum_{i=0}^{m-1} e_i \times (m + 1)^i \quad 0 \leq N \leq (m + 1)^m - 1 \]

The number of required nodes for building the fake watermark is listed in table 5-1.

For the fake watermark with multi-constants, the number of embedded fake watermark and the anti-attack capability of watermark are in direct ratio. However, in table 5-2, it can be seen that while the fake watermarks are constructed, the required space of running program is overloaded. Obviously, it affects the efficiency of program. For this reason, how to balance the intensity of anti-attack and the efficiency of program is one important question to be worthy of note.

VI. EXPERIMENT ANALYSIS AND CONCLUSION

In order to verify the proposed solution, a series of experiment are performed.

The software watermark maybe suffers the different attack such as code confusion and code optimization[6]. The factor of robustness in running period is a key measurement indicator associated with system vulnerability[3]. We hope that, during the tests of the proposed method, the robustness factor is close to 1.

With the bytecode confusion of Java – JBPOT as intimidation model, there are confusion attacks to the DGW subsystem embedded by watermark program. The parameters in tests are listed in table 5-2. (The running time is obtained by averaging the result of 12 different tests).

The results show that, with DGW to embed the watermark, the maximum value of robustness measure is 1.14 and the minimum is 1.01 (close to 1) for radix-k Circular Linklist and IPPCT which were attacked by JBPOT confusion. We can draw a conclusion that IPPCT is robust to this confusion attack and it is a good watermark structure.

This paper proposed an improved watermark structure IPPCT on the basis of PPCT. Both the theories and experiments show that it is superior to the classical radix-k Circular Linklist and PPCT in terms of ability of anti-attack and data ratio. It is an efficient software watermark structure.

Furthermore, combined with the idea of anti-sophisticating, a DGW software fake watermark solution on multi-constants and anti-sophisticating is proposed. The theory analysis and the results of experiments demonstrated that it not only can improve the dynamic hiding ability and anti-attack ability but also plays a key role in tamper proofing.

REFERENCES


[3] C.Collberg and J.Thomborson. Software watermarking: models and theory analysis and the results of experiments demonstrated that it not only can improve the dynamic hiding ability and anti-attack ability but also plays a key role in tamper proofing.


Table 5-2 The robustness analysis of DWG ) W: watermark; N: nodes

<table>
<thead>
<tr>
<th>Test program</th>
<th>Embed watermark using DGW subsystem</th>
<th>Radix-k linked List(W:4,N:6)</th>
<th>PPTC structure(W:1,N:6)</th>
<th>IPPCT structure(W:1,N:6)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exe time after code confusion attack(MS)</td>
<td>The factor of robustness in running period</td>
<td>Exe time after code confusion attack(MS)</td>
<td>The factor of robustness in running period</td>
</tr>
<tr>
<td>DateRate_curve</td>
<td>254</td>
<td>1.02</td>
<td>161</td>
<td>1.03</td>
</tr>
<tr>
<td>Factorial</td>
<td>198797</td>
<td>1.04</td>
<td>340915</td>
<td>1.09</td>
</tr>
<tr>
<td>combination</td>
<td>37047</td>
<td>1.11</td>
<td>21350</td>
<td>1.11</td>
</tr>
<tr>
<td>Keeptime</td>
<td>190</td>
<td>1.01</td>
<td>49</td>
<td>1.03</td>
</tr>
<tr>
<td>drawPosition</td>
<td>143009</td>
<td>1.03</td>
<td>42917</td>
<td>1.04</td>
</tr>
<tr>
<td>valueChanged</td>
<td>191</td>
<td>1.02</td>
<td>175</td>
<td>1.02</td>
</tr>
<tr>
<td>myFrame</td>
<td>981</td>
<td>1.02</td>
<td>1486</td>
<td>1.02</td>
</tr>
<tr>
<td>creatPolyline</td>
<td>194</td>
<td>1.04</td>
<td>197</td>
<td>1.05</td>
</tr>
<tr>
<td>Panel</td>
<td>15842</td>
<td>1.02</td>
<td>5816</td>
<td>1.04</td>
</tr>
<tr>
<td>ActionSelect</td>
<td>7398</td>
<td>5181</td>
<td>1.02</td>
<td>1.02</td>
</tr>
</tbody>
</table>