A Novel Planar IPPCT Tree Structure and Characteristics Analysis

Zhu Jian-qi
College of Computer Science and Technology, Symbol Computation and Knowledge Engineer of Ministry of Education, JiLin University, ChangChun, China
Email: zhujq@jlu.edu.cn

Liu Yan-heng and Yin Ke-xin
College of Computer Science and Technology, Symbol Computation and Knowledge Engineer of Ministry of Education, JiLin University, ChangChun, China
Email: lyh_lk@yahoo.com

College of Computer Science and Engineering, ChangChun University of Technology, ChangChun, China
Email: yinkexin3@hotmail.com

Abstract—A novel planar IPPCT tree (PIPPCT) that holds higher data rate is presented based on analysis of main topologies characteristics of current dynamic graph watermarking system (DGW). Multiple dimensional IPPCT tree (MIPPCT) model and three identity vectors of DataRate, structure and Total concerning it are constructed and presented, then the effect of these vectors on MIPPCT model is discussed in detail. The results proved that the data rate of MIPPCT tree will increase to a limit with its dimension number increasing based on equal number of leaves and this limit is also computed. Moreover, it proved that there exists a two dimensional area HArea, when DataRate is in this area, dynamic graph watermarking system will take on greater performance.

Index Terms—software watermarking, dynamic graph, data rate, IPPCT tree, DataRate, HArea

I. INTRODUCTION

Software watermarking is a tool used to combat software piracy by embedding identifying information into a program to claim the ownership [1-4]. One of the most effective watermarking techniques proposed to date is the dynamic graph watermarking (DGW) scheme of Collberg et al [5], which is resistant to attacks such as optimization and obfuscation. Currently, three main topologies that DGW adopts include Radix-k, PPCT (Planted Plane Cubic Tree) and improved PPCT (IPPCT) structures. Palsberg in his literature [6] not only gives the relationship between the PPCT (Planted Plane Cubic Tree) structure and the integer but also establishes the PPCT enumeration encoding mode. However, the effect is not very good because he can not find out a good way to represent a large number. Wang Yong [7] presents an improved IPPCT, which integrates advantages of Radix and PPCT. IPPCT Not only significantly reduces the space and time complexities but also establishes a valuable software watermarking system that improved the ability to resisting the conspiracy attack.

The goal of this paper is to investigate a practical implementation of dynamic graph watermarking. Focusing particularly on the parameter of data rate, we present two novel dynamic graph watermarking topologies: PIPPCT and MIPPCT structures. The two new topologies have greater performance compared with Radix, PPCT and IPPCT. Lastly, general principles are given on how to choose a MIPPCT tree with higher performance.

II. THE CHARACTERISTICS OF TYPICAL DWG

Fig.1 is the Radix-k graph in a circular linked list of length n. The data pointer field encodes a base-n digit in the length of the path from the node back to itself. A null-pointer encodes a 0, a self-pointer a 1, a pointer to the next node encodes a 2, etc. Fig.2 illustrates a PPCT that is essentially a binary tree with one extra node called origin, which has a pointer to the root of the binary tree. Moreover, all leaves are linked into a circular linked list which includes the origin. Each leaf has a self-pointer and every node in a PPCT has two outgoing pointers. Fig.3 illustrates an improved PPCT (IPPC) which is a Radix encoding scheme based on PPCT.

A good representation for a large number should satisfy the followings [7].

1. High efficiency: the proportion of the number of nodes in the topology graph to the large number should be as small as possible.
2. Continuous representation: no matter what method, it must express continuous integers in a particular scope without vacancy.
3. Time complexity of extraction: the time complexity is O (an+b), n is the number of graph nodes, a and b are constants.
4. For the same number, there should have the greatest possible number of ways that express a large number in order to resist the conspiracy attack.
For any software watermarking system, we should embed the greatest possible number of watermark information on the premises of the program size increasing as small as possible and the resources it occupied as little as possible. The concept of data rate can describe this trait. Ref. [5] gives its definition:

Suppose \( w \in \mathcal{W} \) is the watermark to be embedded and \( H (w) = w \) is the information source entropy (bit). \(|P|\) is the program size of \( P \) and the unit is two byte. \(|I\ell| = \text{Max} \{ \text{dom}(p) | S(p, I,) \} \) is the maximum resources occupied by \( P \) when the input sequence \( I \) has different values and the unit is two bytes. Then the static embedding rate in the program \( P \) is:

\[
\frac{H(w)}{\max(1, |P|)} \geq 1
\]  

(1)

The dynamic embedding rate in the program \( P \) is:

\[
\frac{H(w)}{\max(1, |S(P)|)} \geq 1
\]  

(2)

The range of \( \text{Radix-k} \) is \( 0 \leq n \leq (m+1)^m - 1 \). So the information source entropy is \( \log_2 |w| = \log_2 (m+1)^m \). Suppose DGW system couldn’t bring any other load to the program, according to formula (2), the data rate of DGW system by \( \text{Radix-k} \) is \( \log_2 (m+1)^m / (2m + 1) \approx \log_2 (m+1) / 2 \), and \( m \) equals to \( k-1 \). According to Catalan number theory, the expression range of a PPCT which has \( m \) nodes is \( 0 \leq n \leq \frac{1}{m} C_{m-1}^{2m-2} - 1 \), the source entropy is \( \log_2 |w| = \log_2 C_n^{2m-2} \) and the data rate is \( \log_2 C_n^{2m-2} / 4m \). When \( m \) is big enough, the data rate is about 0.5.

From above we can see that \( \text{Radix-k} \) has higher data rate and less error-detection capability, because their data pointers are less constrained. However, PPCT structure has advantages of both binary tree and chain list, which makes it have better robustness, e.g. When the pointers of some nodes are modified, we can still resume it effectively based on its characteristics. However, its data rate is much lower (≈ 0.5).

Paper [7] presents an improved PPCT structure (IPPCT). The essence of IPPCT is the \( \text{Radix-k} \) that is based on PPCT structure.

The expression range of IPPCT with \( m \) leaf nodes is \( 0 \leq n \leq (m+1)^m - 1 \), the information source entropy is \( \log_2 |w| = \log_2 (m+1)^m \) and the data rate is \( \log_2 (m+1)^m / 4m \approx \log_2 m / 4 \).

The time complexity of watermark recognition algorithm in the literature [7] is \( O(m^2) \). We can have an applicable and big software watermark database by choosing any one from PPCTs (the number of PPCTs is \( C_{m-1}^{2m-2} / m \)) as the carrier of the watermark number, which effectively improved the ability of resisting conspiracy attack.

III. STRUCTURE AND CHARACTERISTICS OF PIPPCT

In order to improve the performance of IPPCT structure, this paper presents a better PIPPCT structure. Fig.4 is a PPCT structure with three leaf nodes, Fig.5 is the corresponding two dimensional PPCT structure.

As Fig.4-(1), the leaf of PPCT has only two points: \( L \) and \( R \), \( R \) points to itself. In Fig.4-(2), the leaf of two dimensional PPCT has three points: \( L, R1 \) and \( R2 \). \( R1 \) and \( R2 \) point to itself, respectively. With \( \text{Radix-k} \), two dimensional PPCT can build PIPPCT. A new domain \( L_a \) (in Fig.5) is added to the leaf of the two dimensional PPCT in order to satisfy the condition 2 in section II.

When an IPPCT with \( m \) leaf nodes encodes, the node number that participates encoding is only \( m \) and the base number is \( m+1 \). However, in PIPPCT case, besides these \( m \) nodes, the \( R2 \) domain in the origin also encodes (\( R1 \) is only used to maintain the topology structure), the base number is \( m + (m+1) - 2m + 2 \). The formula (3) is used to represent the watermark number \( N \).

\[
N = \sum_{i=0}^{m-1} e_i (2m + 2)^i + \sum_{i=m}^{2m} e_i (2m + 2)^i
\]  

(3)

Where, \( e_i \) and \( e_i \) are calculated by formula (4). From the leaf nodes pointed by pointer \( L \) of origin, we use \( L_i \) (\( i \in \{1, 2, ..., m\} \)) to denote \( m \) leaf nodes, respectively. Here we define that the power of each \( L_i \) includes \( i_1 \) and \( i_2 \), \( i \in \{1, 2, ..., m\} \) denotes the power of \( 0 \sim m-1 \) for the base number respectively and \( i_1 = i, i_2 = i + m + 1 \).
From (4), $e_u$ is the number of the leaves from pointer $L$ of leaf node $L_i$ to the leaf node $L_j$ pointed by $R_j$. If pointer $R_u$ is null, $e_u$ is 0. If $R_u$ points to itself, $e_u$ is 1. If $R_u$ points to the next leaf, $e_u$ will be 2 and so on.

According to (3) and (4), the expression range of PIPPCT with $m$ leaf nodes is $2^{10 - 2m} - 1$. It is clear that the data rate of PIPPCT is higher than IPPCT. When the number of leaves is bigger, the higher part is about 0.25. According to formula (3) and (4), this paper realizes a DGW scheme that uses the PIPPCT structure. It is clear that the time complexity in the worst case is $O(m^2)$ and $m$ is the number of leaves of PIPPCT. PIPPCT with $m$ leaf nodes can have $\binom{2m+2}{m+1} - 1$ kinds of topologies, from which we can choose any one as the watermark carrier in order to have a larger software data base that satisfies the conditions 3 and 4 in section II.

This paper evaluates the performances of DGW schemes that are based on PPCT, IPPCT and PIPPCT structures, respectively. The benchmark applications are TTT, calculator and mine-sweeping as shown in Table II and Table III.

| Table I. COMPARISON OF EXPRESSION ABILITY OF IPPCT AND PIPPCT |
|------------------|------------------|
| Leaf number $m$  | IPPCT            | PIPPCT           |
| 1                | $1.0 \times 10^9$| $6.3 \times 10^7$|
| 10               | $2.6 \times 10^{10}$ | $6.0 \times 10^{10}$ |
| 100              | $2.7 \times 10^{200}$ | $2.4 \times 10^{461}$ |

From Table I we can see that the expression ability of PIPPCT is much better than IPPCT when the leaves are same and PIPPCT satisfies the condition 2.

Suppose $L_1, L_2$ are short integer, then the word number of each leaf is $1+2+2 \times 16/32=4$ and the information source entropy of IPPCT with $m$ leaves is $\log_2\left(\frac{(2m+2)(2m+1)}{8m}\right)$, the data rate is:

$$\log_2\left(\frac{(2m+2)(2m+1)}{8m}\right) / (4 \times 2m) \approx (2m+1)\log_2\left(\frac{2m+2}{8m}\right) / 4 \approx \frac{1}{4}$$

Fig.7 illustrates the data rates of IPPCT and PIPPCT. It is clear that the data rate of PIPPCT is higher than IPPCT. When the number of leaves is bigger, the higher part is about 0.25. According to formula (3) and (4), this paper realizes a DGW scheme that uses the PIPPCT structure. It is clear that the time complexity in the worst case is $O(m^2)$ and $m$ is the number of leaves of PIPPCT. PIPPCT with $m$ leaf nodes can have $\binom{2m+2}{m+1} - 1$ kinds of topologies, from which we can choose any one as the watermark carrier in order to have a larger software data base that satisfies the conditions 3 and 4 in section II.

<table>
<thead>
<tr>
<th>Table II. BENCHMARK APPLICATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programs</td>
</tr>
<tr>
<td>TTT</td>
</tr>
<tr>
<td>Calculator</td>
</tr>
<tr>
<td>Mine-sweeping</td>
</tr>
</tbody>
</table>

| Table III. CODE CHARACTERISTICS OF THE THREE STRUCTURES |
|-----------------|------------------|------------------|
| Graph structures | Static code characteristics | Code characteristics of most common 2-grams |
| Number instructions | Number instruction types | Number instructions | Number instruction types |
| PPCT             | 1161             | 50              | 1123     | 240     |
| IPPCT            | 2209             | 59              | 2153     | 326     |
| PIPPCT           | 4438             | 61              | 4325     | 336     |

| Table IV. INSTRUCTION FREQUENCIES OF MOST COMMON 1-GRAMS |
|-----------------|-----------------|
| Opcode           | %               |
| PPCT             | IPPCT           |
| aload           | 24.46%          | 25.12%          | 25.06%  |
| invokevirtual    | 8.53%           | 8.33%           | 8.31%   |
| astore           | 6.03%           | 7.74%           | 7.75%   |
| iconst           | 5.86%           | 6.84%           | 6.83%   |
| getfield         | 4.91%           | 5.16%           | 5.14%   |
| iload            | 4.74%           | 4.84%           | 4.82%   |
| invokespecial    | 4.39%           | 4.12%           | 4.12%   |
| invokeinterface  | 4.05%           | 3.35%           | 3.36%   |

© 2010 ACADEMY PUBLISHER
new 3.45% getfield 2.9% getfield 2.88%
dup 3.45% new 2.72% dup 2.75%
goto 3.1% dup 2.72% new 2.73%
invokestatic 2.76% istore 2.49% istore 2.48%
inc 1.89% invokestatic 2.26% invokestatic 2.25%
areturn 1.46% inc 1.54% inc 1.53%
return 1.38% areturn 1.18% areturn 1.17%
putfield 1.38% aaload 1.09% aaload 1.08%
ldc 1.29% if_icmplt 1% if_icmplt 0.99%
pop 1.12% arraylength 1% arraylength 0.99%
athrow 1.03% return 0.95% return 0.97%
getstatic 1.03% pop 0.91% pop 0.92%
ifeq 1.03% checkcast 0.86% checkcast 0.86%

Table V.

<table>
<thead>
<tr>
<th>Obfuscators</th>
<th>TTT</th>
<th>Calculator</th>
<th>Mine- sweeping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array Folder</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Array Splitter</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>BLOAT</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Block Marker</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Bludgeon Signatures</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Boolean Splitter</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Branch Inverter</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Buggy Code</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Class Encrypter</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Class Splitter</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ConstantPoolReorderer</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Duplicate Registers</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Dynamic Inliner</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>False Refactor</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Field Assignment</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Inliner</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>InsertOpaquePredicates</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Integer Array Splitter</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Interleave Methods</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Irreducibility</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Table VI.

<table>
<thead>
<tr>
<th>Optimizers</th>
<th>TTT</th>
<th>Calculator</th>
<th>Mine- sweeping</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOAT</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Dynamic Inliner</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Inliner</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Reassigner</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Table III includes the static (1-grams) and dynamic (2-grams) code characteristics of PPCT, IPPCT and PIPPCT. Table IV is the instruction frequencies of the three structures. Collberg [9] analyzes the static Java byte code of a sample of 1132 programs and concludes that the frequencies of instructions such as `aload`, `invokespecial`, `getfield`, `dup`, `invokespecial` and so on in a general program are higher. Especially, the frequency of instruction `aload` is the highest (about 19.8%) , most other instruction frequencies are lower than 1% , and the instructions `jsr w` and `goto w` are seldom used. As in Table IV, the frequency of `aload` in PIPPCT and PPCT is higher. Compared to [9], the stealthy of both structures are general and PIPPCT is declined slightly.

This paper studies the watermark schemes of PPCT, IPPCT and PIPPCT structures with optimizers and obfuscators in Sandmark. After the watermark is embedded into the benchmark application, the optimizers or obfuscators in Sandmark begin to attack, and then we extract the watermark. This procedure is called as an anti-attack experiment. Here, we assume that the watermark algorithm can effectively resist this kind of attack only if the watermark is successfully extracted for a total of 10 times after successive anti-attack experiments and we use “+” to represent it. If there is a total of 3 times for failure watermark extraction, we believe that the algorithm can’t resist this kind of attack, and “-” to represent it. If the result is uncertain, we neglect it and continue. The experiment results are as Table V and Table VI. It shows that the anti-attack performance of PPCT, IPPCT and PIPPCT are good and same basically. Moreover, it is clear that PIPPCT inherits the stability of PPCT binary tree structure, and the anti-attack performance of PIPPCT is the same as PPCT, which can effectively resist the semantic-keeping transform in Sandmark and can be immune to all the obfuscators except Split Classes.

To sum up, PIPPCT structure not only inherits the binary tree characteristic of PPCT but also has the high encoding rate feature. Moreover, PIPPCT satisfies the condition 1-4 and has roughly the same stealthy,
robustness and anti-attack performances as PPCT so that PIPPCT has higher overall performance.

IV. MIPPCT Structure Model

We expand PIPPCT model to build MIPPCT structure model. The domains of each node include \( L, R_1, R_2, \ldots, R_n, L_a_1, L_a_2, \ldots, L_a_n \). \( L \) and \( R \) are used to maintain the topology structure and \( R_j \) \((i \in \{1, 2, \ldots, n\})\) is used to encode.

From the leaf node pointed by the pointer \( L \) of origin, we use \( L_j \) \((i \in \{1, 2, \ldots, m\})\) to denote the leaf nodes (the number is \( m \)) respectively and the power of \( 0 \sim m-1 \) for the base number respectively and \( i_1 = i, i_2 = i + m + 1, \ldots, i_n = i + (n-1) \times (m+1) \).

\[
\begin{align*}
e_{j_i} & = \left\{ \begin{array}{ll}
(v(m+1)) \text{, } & \text{if } R_j = \text{NULL} \\
v - j + i + v(m+1) \text{, } & \text{if } j \geq i \\
 m - j - i + v(m+1) & \text{, if } j < i
\end{array} \right.
\end{align*}
\]

\( u \in \{1, 2, \ldots, n\}, v = \text{Value}(L_a_u) \in \{0, 1 \ldots n-1\} \) \((5)\)

That is to say, \( e_{j_i} \) is the number of the leaf nodes from pointer \( L \) to the leaf node pointed by \( R_j \). If pointer \( R_j \) is null, \( e_{j_i} = 0 \). If \( R_j \) points to itself, \( e_{j_i} = 1 \). If \( R_j \) points to the next leaf, \( e_{j_i} = 2 \) and so on. \( L_a_u \) is the circle number that \( R_j \) passes by.

The expression range of MIPPCT with \( m \) leaves is:

\[
N = \sum_{j=0}^{m} e_{j_0} + \sum_{i=1}^{m} e_{j_i} + \sum_{i=1}^{m-1} e_{j_i} (m+n)^i\cdot
\]

Where, \( 0 \leq e_{j_i} < nm+n \). We can get \( 0 \leq N \leq (nm+n)\cdot(\text{nm}+n-1) \) easily.

\( R_i, L_a_u \) and the bit number of \( L_a_u \) will increase with \( n \) increasing. Assume \( L_a_u \) can be allotted according to bit, then the number of double word of each \( L_a_u \) and each node are \([\log_2(m+1)]/32\) and \([\log_2(m+n+1)]/32\) respectively. The information source entropy is \( \log_2 = \log_2(m+n+1) \) and the data rate is:

\[
\text{DataRate}_{\text{source}} = (\log_2(m+n+1))/((1+n+n+2)\times(\log_2(m+1))/32) \times 2|c| \quad (6)
\]

We define a group of vectors in order to analyze the inherent laws of MIPPCT model further. These vectors are data rate \( \text{DataRate} = (n,m) \), topology structure \( \text{Structure} = (m,c) \) and total vector \( \text{Total} = (n,m,c) \). In these vectors, \( n \) denotes the dimension of MIPPCT structure, \( m \) denotes the number of leaf nodes and \( c \) is Catalan index. YongHe gives the conversion formula between PPCT and \( c \) in his literature \([10]\). According to YongHe’s result, we give the conversion formula between MIPPCT and \( c \):

\[
\begin{align*}
\text{min}_\text{int}(|T.L|, |T.R.|) & = \text{min}_\text{int}(|T.L| - 1, |T.R.| + 1) + c(|T.R.| - 1) \times c(|T.R.| + 1) \\
(L \neq 1)
\text{min}_\text{int}(1, R_j) & = 0 \\
\text{int}(T) & = \text{int}(T.L) \times c(\text{LeafNum}(T.R.i)) \\
& + \text{int}(T.R.i) \\
& + \text{min}_\text{int}(\text{LeafNum}(T.L), \text{LeafNum}(T.R.i))
\end{align*}
\]

\( \text{int}(Leaf) = 0 \)

Where, \( \text{int}(T) \) is called the Catalan index number of MIPPCT and represented by \( c \).

Algo 1. Construct an \( n \)-dimensional MIPPCT with \( m \) leaf nodes and make the Catalan index number \( C \):

1. \( \text{MIPPCT\_Node encode}(\text{int } m, \text{ int } C) \)
2. \( \text{int } L, R \)
3. if ((C >= C(m)) return null; // watermark number \( N \) is too big so that \( n \)-dimensional MIPPCT can not express it;
   end if;
4. else if (C == 0 \&\& m == 1) //construct and return an \( n \)-dimensional MIPPCT with only one leaf node and the Catalan index number is 0;
5. else if (C == 0 \&\& m == 2) //construct and return an \( n \)-dimensional MIPPCT with only two leaf nodes and the Catalan index number is 0;
6. else if (C < 3) return null;
   end if;
   end if;
7. \( L = 1; \)
8. while (C(L) <= i) L++;
   end while
9. \( R = n - L; \) // (l-1) leaf nodes are in the left subtree and (R+1) leaf nodes are in the right subtree;
10. \( \text{MIPPCT\_Node left\_subtree, right\_subtree}; \)
11. \( \text{left\_subtree} = \text{encode}((L-1, C - \text{min}_\text{int} (L - 1, R + 1)) / c(R + 1)); \)
12. \( \text{right\_subtree} = \text{encode}((R+1, C - \text{min}_\text{int} (L - 1, R + 1)) \% c(R + 1)); \)
13. //return an \( n \)-dimensional MIPPCT with left\_subtree and right\_subtree as left subtree and right subtree respectively.
End

MIPPCT structure instances with different dimensions can be get by concreting \( n \).

Definition 1.
MIPPCT instances with different dimension \( n \) but having the same topology are called MIPPCT isomorphism.

Definition 2.
MIPPCT instances with the same data rate are called MIPPCT isorate.

Theorem 1.
MIPPCT instances with the same \( \text{Structure} \) are MIPPCT isomorphism.
Proof: Assume two MIPPCT instances with dimensions of \( n_0 \) and \( n_1 \) having the same structure, then they have the same number of leaf nodes and the Catalan index number. According to Algo 1, there will be two MIPPCT structures with the same topologies. Qed.

**Theorem 2.**
MIPPCT instances with the same DataRate are isorate.
Proof: Theorem 2 is clearly correct according to formula (6). Qed.

**Theorem 3.**
MIPPCT instances with the same DataRate will have the same space complexities.
Proof: Assume two MIPPCT instances with the same DataRate (which means they have the same number of leaf nodes \( m \) and dimension \( n \)), and then they will have the same number of nodes and the same node structure. That is to say, the space consumptions occupied by these two instances are same. Qed.

V. CHARACTERISTICS ANALYSIS OF MIPPCT

We define a function
\[
f(x) = \log_2 \left( \frac{x^{m+1} + x^{m+2}}{m+x} \right) / (\log_2 x) + \left( \log_2 \left( \frac{x}{32} \right) \right) \times 2m,
\]
then
\[
\lim_{x \to \infty} f(x) = \lim_{a \to \infty} \log_{a+1}^{(m+1)(a+2)} / (1+x) = 16 \times (1 + \frac{1}{m}).
\]
Sequence can be seen as a special limit and when \( n \to \infty \), DataRate of MIPPCT instances with the same structure is increasing and inclined to a value. The performance of DGW system (robustness and stealthy) will be affected greatly and even declined when \( n \) is increasing to some extent. On one hand, the increasing of \( n \) makes the space of every node occupies bigger, so, we must adopt a topology with lesser nodes in order to decrease the space complexity; however, which decreases the ability of resisting conspiracy attack for DGW system. On the other hand, the augment of node structure make it easier to be located, and then it is not difficult to be attacked by crop and distort attacks. In order to decrease above affects, the literature [11] presents a method of introducing the fake watermarks that are relative to the program correctness. Thus, the parameters of data rate, robustness and stealthy must be considered synthetically in order to achieve optimal performance for DGW system rather than blindly increase the dimension \( n \).

We define a function
\[
F(x, y) = \log_2 \left( y^{x+y} x^{y+1} \right) / (1+x) + \left( \log_2 x \right) \times 2y - (x is fixed). When \( y \to \infty \), \( F(x, y) \) is second-order derivative, there exist a maximum \( \max_F \) and a minimum \( \min_F \) in this area, and
\[
\max_F = \max \{ F(x, y) \mid (x, y) \in HArea \}
\]
\[
\min_F = \min \{ F(x, y) \mid (x, y) \in HArea \}
\]
Here, \( HArea \) is called as high power region of MIPPCT structure, it is clear that the values of \( \max_F \) and \( \min_F \) will be affected by various factors such as the robustness or stealthy.

To sum up, the general principles of choosing MIPPCT instance are: we should choose such MIPPCT structure with smaller dimension \( n \) and more leaf nodes \( m \) under the circumstances of being the same system.
space load. On one hand, the data rate of DGW system is generally higher; on the other hand, we can get more MIPPCT topologies when the leaf node \( m \) is bigger, which is favor of withstanding the conspiracy attack. Moreover, smaller \( n \) will help improve the watermark graph stealthy. Because the watermark number \( N \) is corresponding to a MIPPCT structure, a high performance tree structure requires \( N \) not being too big. The literature [12] presents a technique of watermark sharing, which improves the DGW system performance with an appropriate size of watermark number. Table VIII shows a comparative analysis of various DGW topologies.

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>Data rate</th>
<th>Robust</th>
<th>Stealth</th>
<th>Time complexity</th>
<th>Space Complexity</th>
<th>Overall Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radix-k</td>
<td>( \log_2 m / 2 )</td>
<td>bad</td>
<td>good</td>
<td>( O(m^2) )</td>
<td>good</td>
<td>moderate</td>
</tr>
<tr>
<td>PPCT</td>
<td>( \log_2 m / 4 )</td>
<td>good</td>
<td>good</td>
<td>( O(m^2) )</td>
<td>normal</td>
<td>moderate</td>
</tr>
<tr>
<td>IPPCT</td>
<td>( \log_2 m / 4 )</td>
<td>good</td>
<td>good</td>
<td>( O(m^2) )</td>
<td>normal</td>
<td>good</td>
</tr>
<tr>
<td>PIPPCT</td>
<td>( \log_2 m / 4 + 0.25 )</td>
<td>good</td>
<td>good</td>
<td>( O(m^2) )</td>
<td>good</td>
<td>excellent</td>
</tr>
<tr>
<td>MIPPCT</td>
<td>( 10^{(1 + 1 / m)} )</td>
<td>good</td>
<td>Decline with ( m )</td>
<td>( O(m^2) )</td>
<td>excellent</td>
<td>excellent</td>
</tr>
</tbody>
</table>

VI. SUMMARIES AND OUTLOOK

This work first analyzes the structural characteristics of the current mainstream DGW topologies such as Radix-k, PPCT and IPPCT. In particular, we study the data rate attribute and pointed out that the data rate of Radix-k is higher (\( \log^m / 2 \)) but the structure is simple and having weak error-correcting ability. On the contrary, PPCT has a better tamper-proofing ability and robustness but its data rate is lower (\( \approx 0.5 \)). IPPCT structure that has a better tamper-proofing ability, actually, is a kind of Radix encoding scheme based on PPCT with an improved data rate (\( \log^m / 4 \)).

Secondly, this paper presented an improved PIPPCT structure with a greater increase in data rate (\( \log^m / 4 + 0.25 \)) than IPPCT. Then, we bring forward the MIPPCT structure model, the characteristic vectors ((DataRate), (structure), (Total)) and H\( \alpha \)ea concepts. We analyze the changing rules of the model performance with these vectors and point that there is an area H\( \alpha \)ea in which MIPPCT has a higher performance. Lastly, the general principles of choosing a MIPPCT with higher performance are given.

There are still many problems before MIPPCT structure is applied in practice. For example, we should firstly consider the factors such as the data rate, stealthy and robustness synthetically and make sure the boundary of H\( \alpha \)ea. Then, in order to improve the tamper-proofing ability of DGW system, we should take some areas as the tamper-proofing encoding areas in the multi-dimension structure.

ACKNOWLEDGMENT

This work is supported by both the National Nature Science Foundation of China (No.60573128) and the National Research Foundation for the Doctoral Program of Higher Education of China (No.20060183043).

REFERENCES

ZHJU Jian-qi was born in Zhenjiang, China, in 1976. He received the M.S. degree in computer application from Jilin University, Jilin, China, in 2004. Since 1999, he has been working in the department of computer science and technology, Jilin University. He is currently working toward the Ph.D. degree and main areas of interests include software protection, obfuscation, software watermarking, and so on. He published 16 papers in “Computer Science”, “Journal of Jilin University (engineering and technology edition)”, “Journal of computer research and development” and so on, 9 of those were indexed by EI.

LIU Yan-heng was born in Jilin, China, in 1958. He received the Ph.D degree in mobile communication from Jilin University, Jilin, China, in 2002. His primary research fields includes: Network communication and protocol design, QoS mechanism in Mobile IP network, Policy-based network management and network intrusion detection system, and so on. Supported by National study abroad foundation committee, he has been to Motorola Company in Canada (2000.7—2001.4), the University of British Columbia in Canada (1998.8—1999.8) and University of Hull in England (1988.9—1989.12) for science research collaboration and as a visiting scholar. Now he is professor of Jilin University, tutor of Ph. D. student. He has taken charge of 12 research subjects of ministry or province's, taken part in research work in national natural science foundation and 863 subjects, among the production, one is chosen in the “Chinese eighth five years' science research achievements”. As the 1st author, published 33 papers in “Journal of Software”, “Journal of Computer Research and Development”, “Journal of Communication”, 10 of those were indexed by SCI and EI; main editor of 4 textbooks.

YIN Ke-xin was born in Jilin, China in 1975. She received the M.S. degree in computer application from Changchun University of Science and Technology (CUST) in 2004 and the Ph.D. degree in optical communication from CUST in 2008. She has been a teacher in Changchun University of Technology since 2008. Her research interests lie in the information security and digital watermarking and so on. She published 12 papers in “Journal of Jilin University (engineering and technology edition)”, “Chinese Optics Letters” and so on, 6 of these were indexed by EI.