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_lb_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 $\overrightarrow{DataRate}$, $\overrightarrow{structure}$ and $\overrightarrow{Total}$ 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 *n* 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 $\overrightarrow{DataRate}$ is in this area, dynamic graph watermarking system will take on greater performance.**

***Index Terms*—software watermarking, dynamic graph, data rate, PIPPCT tree, $\overrightarrow{DataRate}$, 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 (IPPCT) 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.

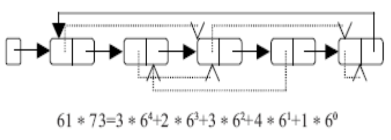


Figure 1. RADIX

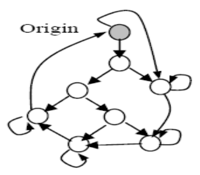


Figure 2. PPCT

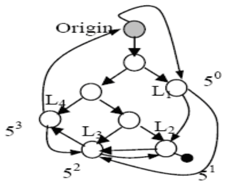


Figure 3. IPPCT

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 $w \in W$ 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. |S(p)|=Max $I \in dom(p)$ |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 Pw is:

$$\frac{H(w)}{\max(1, |P_w| - |P|)} \geq 1 \tag{1}$$

The dynamic embedding rate in the program Pw is:

$$\frac{H(w)}{\max(1, |S(P_w)| - |S(P)|)} \geq 1 \tag{2}$$

The range of Radix-k is $0 \leq n \leq (m+1)^m - 1$. So the information source entropy is $\log_2^n = \log_2^{(m+1)^m}$. Suppose DGW system couldn't bring any other load to the program, according to formula (2), the data rate of DGW system by Radix-k is $\log_2^{(m+1)^m} / (2m+1) \approx \log_2{}^m / 2$, and m equals to k-1. According to Catalan number theory [8], the expression range of a PPCT which has m nodes is $0 \leq n \leq \frac{1}{m} C_{2m-2}^{m-1} - 1$, the source entropy is $\log_2^n = \log_2^{\frac{1}{m} C_{2m-2}^{m-1}}$ and the data rate is $\log_2^{\frac{1}{m} C_{2m-2}^{m-1}} / 4m$ .When m is big enough, the data rate is about 0.5.

From above we can see that 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).

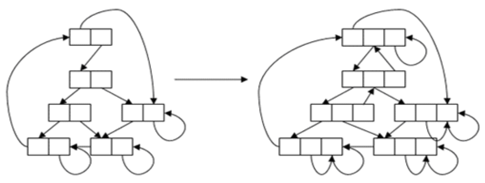
Paper [7] presents an improved PPCT structure (IPPCT). The essence of IPPCT is the Radix-k that is based on PPCT structure.

The expression range of IPPCT with m leaf nodes is $0 \leq n \leq (m+1)^m - 1$, the information source entropy is $\log_2^n = \log_2^{(m+1)^m}$ 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_{2m-2}^{m-1} / 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.



(1) PPCT (2) Two dimensional PPCT

Figure 4. PPCT and two-dimensional PPCT

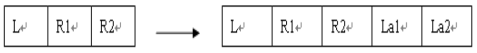


Figure 5. Transform of domain from two-dimensional PPCT to PIPPCT

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 Radix-k, two dimensional PPCT can build PIPPCT. A new domain La_u (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.

$$N = \sum_{i_1=0}^{m-1} e_{i_1} (2m+2)^{i_1} + \sum_{i_2=m}^{2m} e_{i_2} (2m+2)^{i_2} \tag{3}$$

Where, e_{i_1} and e_{i_2} are calculated by formula (4). From the leaf nodes pointed by pointer L of origin, we use L_i (i ∈ {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∈ {1,2,…,m} denotes the power of 0 ~ m-1 for the base number respectively and $i_1 = i$, $i_2 = i+m+1$.

$$e_{i_u} = \begin{cases} v(m+1) & ,R_u \text{ is NULL} \\ |(j-i)+1|+v(m+1) & ,\ j \geq i \\ m-|j-i+1|+v(m+1) & ,\ j < i \end{cases},$$

$$u \in \{1,2\}, v \in Value(La_u) \qquad (4)$$

From (4), e_{i_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_{i_u} is 0. If R_u points to itself, e_{i_u} is 1. If R_u points to the next leaf, e_{i_u} will be 2 and so on.

According to (3) and (4), the expression range of PIPPCT with m leaf nodes is $0 \leq n \leq (2m+2)^{2m+1} - 1$. It is clear that PIPPCT satisfies the condition 1. Fig.6 illustrates a PIPPCT structure with 3 leaves. In this case, k=2m+2=8， N=0$\times 8^6 + 5 \times 8^5 + 7 \times 8^4 + 7 \times 8^3 + 1 \times 8^2 + 7 \times 8^1 + 7 \times 8^0$ = 317×619 = 196223.

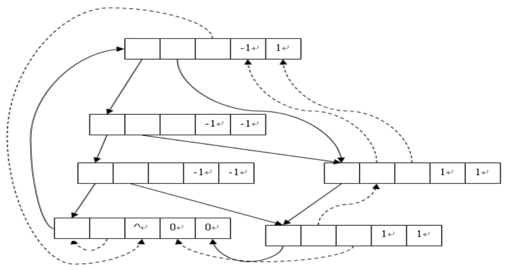


Figure 6. PIPPCT with three leaf nodes

The comparison of expression ability of IPPCT and PIPPCT is shown in Table I.

TABLE I.

COMPARISON OF EXPRESSION ABILITY OF IPPCT AND PIPPCT

Leaf number m	IPPCT	PIPPCT
1	1.0×10^{0}	6.3×10^{1}
10	2.6×10^{10}	6.0×10^{10}
100	2.7×10^{200}	2.4×10^{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 La_1, La_2 are short integer, then the word number of each leaf is 1+2+2 × 16/32=4 and the information source entropy of IPPCT with m leaves is $\log_2^n = \log_2^{(2m+2)^{2m+1}}$, the data rate is:

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

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 C_{2m-2}^{m-1}/m 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.

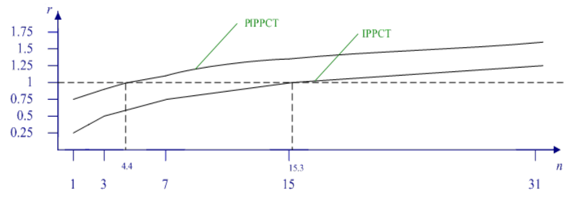


Figure 7. The data rates of IPPCT and PIPPCT

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 II.

BENCHMARK APPLICATIONS

Programs	Number classes	Number methods	Size(KB)
TTT	12	51	11
Calculator	2	6	4
Mine-sweeping	8	38	42

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

PPCT		IPPCT		PIPPCT	
Opcode	%	Opcode	%	Opcode	%
aload	24.46%	aload	25.12%	aload	25.06%
invokevirtual	8.53%	astore	8.33%	astore	8.31%
astore	6.03%	invokevirtual	7.74%	invokevirtual	7.75%
iconst	5.86%	iload	6.84%	iload	6.83%
getfield	4.91%	invokeinterface	5.16%	invokeinterface	5.14%
iload	4.74%	iconst	4.84%	iconst	4.82%
invokespecial	4.39%	goto	4.12%	goto	4.12%
invokeinterface	4.05%	invokespecial	3.35%	invokespecial	3.36%

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%
istore	1.89%	invokestatic	2.26%	invokestatic	2.25%
iinc	1.46%	ifne	1.54%	ifne	1.53%
areturn	1.46%	iinc	1.54%	iinc	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.

THE EFFECTS OF OBFUSCATIONS

Obfuscators	TTT	Calculator	Mine- sweeping	Obfuscators	TTT	Calculator	Mine- sweeping
Array Folder	+	+	+	Merge Local Integer	+	+	+
Array Splitter	+	+	+	Method Merger	+	+	+
BLOAT	+	+	+	Objectify	+	+	+
Block Marker	+	+	+	Opaque Branch Insertion	+	+	+
Bludgeon Signatures	+	+	+	Overlode Names	+	+	+
Boolean Splitter	+	+	+	ParamAlias	+	+	+
Branch Inverter	+	+	+	PromotePrimitive Registers	+	+	+
Buggy Code	+	+	+	Promote Primitive Types	+	+	+
Class Encrypter	+	+	+	Publicize Fields	+	+	+
Class Splitter	+	+	+	Random Dead Code	+	+	+
ConstantPoolReorderer	+	+	+	Rename Registers	+	+	+
Duplicate Registers	+	+	+	Reorder instructions	+	+	+
Dynamic Inliner	+	+	+	Reorder Parameters	+	+	+
False Refactor	+	+	+	Simple Opaque Predicates	+	+	+
Field Assignment	+	+	+	Split Classes	-	-	-
Inliner	+	+	+	Static Method Bodies	+	+	+
InsertOpaquePredicates	+	+	+	String Encoder	+	+	+
Integer Array Splitter	+	+	+	TransparentBranchInsertion	+	+	+
Interleave Methods	+	+	+	VariableReassigner	+	+	+
Irreducibility	+	+	+				

TABLE VI.

THE EFFECTS OF OPTIMIZERS

Optimizers	TTT	Calculator	Mine- sweeping	Optimizers	TTT	Calculator	Mine- sweeping
BLOAT	+	+	+	Inliner	+	+	+
Dynamic Inliner	+	+	+	Variable Reassigner	+	+	+

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*, *invokevirtual*, *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 , $R_1, R_2, \ldots, R_n$, $La_1, La_2, \ldots, La_n$. L and R are used to maintain the topology structure and R_i （$i \in \{1,2,\ldots,n\}$） is used to encode.

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

$$e_{i_u} = \begin{cases} v(m+1) & ,R_u \text{ is NULL} \\ |(j-i)+1| + v(m+1) & , j \geq i \\ m - |j-i+1| + v(m+1) & , j < i \end{cases}$$

$$u \in \{1,2,\ldots,n\}, v = Value(La_u) \in \{0, 1 \ldots n-1\} \quad (5)$$

That is to say, e_{i_j} is the number of the leaf nodes from pointer L to the leaf node pointed by R_j . If pointer R_u is null, e_{i_u} is 0. If R_u points to itself, e_{i_u} is 1. If R_u points to the next leaf, e_{i_u} will be 2 and so on. La_u is the circle number that R_u passes by.

The expression range of MIPPCT with m leaves is:

$$N = \sum_{i_1=0}^{m-1} e_{i_1}(nm+n)^{i_1} + \sum_{i_2=m}^{2m} e_{i_2}(nm+n)^{i_2} + \ldots + \sum_{i_n=(n-1)(m+1)-1}^{nm+n-2} e_{i_n}(nm+n)^{i_n}$$

Where, $0 \leq e_{i_u} < nm+n$. We can get $0 \leq N \leq (nm+n)^{nm+n-1} - 1$ easily.

R_i , La_u and the bit number of La_u will increase with n increasing. Assume La_u can be allotted according to bit, then the number of double word of each La_u and each node are $\left\lceil \left\lceil \log_2^n \right\rceil / 32 \right\rceil$ and $1 + n + n \times \left\lceil \left\lceil \log_2^n \right\rceil / 32 \right\rceil$ respectively. The information source entropy is $\log_2^n = \log_2^{(nm+n)^{nm+n-1}}$ and the data rate is:

$$DataRate_{MIPPCT} = (\log_2^{(nm+n)^{nm+n-1}}) / ((1+n+n \times \left\lceil \left\lceil \log_2^n \right\rceil / 32 \right\rceil) \times 2m) \quad (6)$$

We define a group of vectors in order to analyze the inherent laws of MIPPCT model further. These vectors are data rate $\overrightarrow{DataRate} = \overrightarrow{(n,m)}$, topology structure $\overrightarrow{Structure} = \overrightarrow{(m,c)}$ and total vector $\overrightarrow{Total} = \overrightarrow{(n,m,c)}$. 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:

$$\begin{aligned} \min\_\text{int}(|\text{T.L}|, |\text{T.R}_1|) = {} & \min\_\text{int}(|\text{T.L}|-1, |\text{T.R}_1|+1) \\ & + \text{c}(|\text{T.R}_1|-1) \times \text{c}(|\text{T.R}_1|+1) \end{aligned}$$

$$(\text{L} \neq 1)$$

$$\min\_\text{int}(1, \text{R}_1) = 0$$

$$\begin{aligned} \text{int}(T) = {} & \text{int}(T.L) \times c(LeafNum(T.R_1)) \\ & + \text{int}(T.R_1) \\ & + \min\_\text{int}(LeafNum(T.L), LeafNum(T.R_1)) \end{aligned}$$

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

Where, $\text{int}(T)$ is called the Calalan 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． MIPPCT_Node encode(int m, int C)
2．   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．   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;
    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．   MIPPCT_Node  left_subtree, right_subtree;
11．   left_subtree = encode( (L-1, C - min_int( L – 1 , R
+ 1 )) / c(R + 1) );
12．   right_subtree = encode((R+1, C - min_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 $\overrightarrow{Structure}$ are MIPPCT isomorphism.

Proof: Assume two MIPPCT instances with dimensions of n_0 and n_1 having the same $\overrightarrow{Structure}$, 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 $\overrightarrow{DataRate}$ are isorate.

Proof: Theorem2 is clearly correct according to formula (6). Qed.

Theorem 3.

MIPPCT instances with the same $\overrightarrow{DataRate}$ will have the same space complexities.

Proof: Assume two MIPPCT instances with the same $\overrightarrow{DataRate}$ (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

$$f(x) = \log_2^{(xm+x)^{xm+x-1}} / ((1+x+x\times\lceil\lceil\log_2^{x}\rceil/32\rceil)\times 2m) \quad ,$$

then

$$\lim_{x\to\infty} f(x) = \lim_{x\to\infty} \log_2^{(xm+x)^{xm+x-1}} / ((1+x+x\times\log_2^{x}/32)\times 2m) = 16\times(1+\frac{1}{m})$$

. Sequence can been seen as a special limit and when $n\to\infty$, $DataRate_{MIPPCT}$=f(*n*)=$16\times(1+\frac{1}{m})$. Table VII shows the data rate distributions of MIPPCT structure with different dimensions.

When the number of leaf nodes *m* is 3, 7 and 31, the values of $DataRate_{MIPPCT}$ is shown in Fig.8. We can see that the increasing rate of $DataRate_{MIPPCT}$ is greater when *n* is small and *m* is fixed. When *n* is bigger, the increasing rate of $DataRate_{MIPPCT}$ is smaller and if $n\to\infty$, $DataRate_{MIPPCT}$ =16(1+ $1/m$). Moreover, $DataRate_{MIPPCT}$ will increase with the number of leaf nodes *m* increasing especially when *n* is smaller.

TABLE VII.
DATA RATE OF MIPPCT

m \ n	1	2	4	8	16	32	2^{32}	$+\infty$
m=3	0.33	1.14	1.90	2.65	3.32	3.90	11.33	21.33
m=7	0.75	1.4	2.11	2.77	3.34	3.83	10	18.29
m=31	1.25	1.99	2.73	3.37	3.90	4.34	9.55	16.52

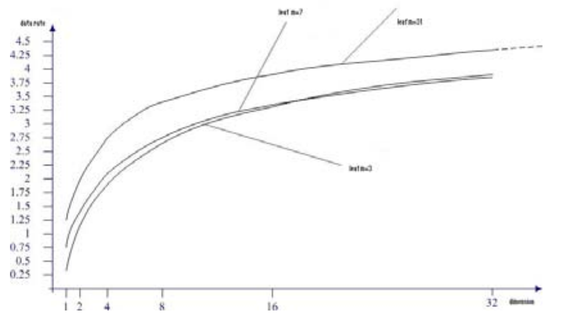


Figure 8. Data rate of MIPPCT

From formula (6), we can get that with *n* increasing, $DataRate_{MIPPCT}$ 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 dimension *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 augments 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 decease 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

$$F(x,y)=(\log_2^{(xy+x)^{xy+x-1}})/(1+x+x\times\lceil\lceil\log_2^{x}\rceil/32\rceil)\times 2y \quad (x \text{ is}$$

fixed). When $y\to\infty$, $F(x, y)=A\times\log_2^{y+1}+B$, and $A=\dfrac{x}{1+x+x\log_2^{x}/32}$, $B=\dfrac{x\log_2^{x}}{1+x+x\log_2^{x}/32}$. That is, $F(x, y)$ will increase at the magnitude of O $(\log^{y})$.

We predict that there exists a two-dimensional regional *HArea*, when $\overrightarrow{DataRate}\in HArea$, DGW system has better performance. As $F(x, y)$ is second-order derivative, there exist a maximum $f_{\max}$ and a minimum $f_{\min}$ in this area, and

$$f_{\max} = \max\{F(x,y) \mid (x,y)\in HArea\}$$

$$f_{\min} = \min\{F(x,y) \mid (x,y)\in HArea\}$$

Here, *HArea* is called as high power region of MIPPCT structure, it is clear that the values of $f_{\max}$ and $f_{\min}$ 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 VIII.

COMPARIONS OF VARIOUS DGW TOPOLOGIES

attri bute s / DGW	data rate	robust	stealth	time complexity	space complexity	overall performance
Radix-k	$\log_2^m/2$	bad	good	O(m^2)	good	moderate
PPCT	≈0.5	good	good	O(m^2)	normal	moderate
IPPCT	$\log_2^m/4$	good	good	O(m^2)	normal	good
PIPPCT	$\approx\log_2^m/4+0.25$	good	good	O(m^2)	good	excellent
MIPPCT	$16(1+1/m)$	good	Decline with *n* increase	O(m^2)	excellent	excellent

VI. SUMMARIES AND OUTLOOK

This work first analyzes the structure 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 (≈ 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 ($\log_2^m/4+$ 0.25) than IPPCT. Then, we bring forward the MIPPCT structure model, the characteristic vectors (($\overrightarrow{DataRate}$), ($\overrightarrow{structure}$), ($\overrightarrow{Total}$)) and *HAea* concepts. We analyze the changing rules of the model performance with these vectors and point that there is an area *HAea* 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 *HAea*. 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

[1] Zhang Li-He , Yand Yi-Xian, Niu Xin-Xin, NIU Shao-Zhang , A Survey on Software Watermarking [J], Journal of Software, 2003,14(2):268-277.
[2] W.Zhu,C.Thomborson,F.-Y.Wang.A surveyof software watermarking.In IEEE ISI 2005,volume 3495 of LNCS, pp.454–458,May 2005.
[3] W.Zhu,C.Thomborson.Recognition in Software Watermarking.in 1st ACM Workshop on Multimedia Content Protection and Security,in conjunction with ACM Multimedia 2006, October 27th, 2006, Santa Barbara, CA, USA, pp.29-36.
[4] W.Zhu,C.Thomborson.Extraction in Software Watermarking.in ACM MM&Sec 06 Workshop,26-27,Sept,2006,Switzerland,pp.175--181.
[5] C.Collerg ,C.Thomborson.Software Watermarking:Models and Dynamic Embeddings, in'Principles of Programming Languages(POPL'99)',San Antonio,TX,pp.311-324,1999.
[6] J.Palsberg,S.Krishnaswami,M.Kwon,D.Ma, Q.Shao & Y.Zhang .Experience with software watermarking,in'Proc.16th Ann.Comp.Security Application Conf.(ACSAC'00)',IEEE Computer Society,pp.308-316,2000.
[7] WangYong, Yang Yixian. A software watermark database scheme based on PPCT [C]. CIHW2004, 2004.
[8] I.P.Goulden ,D.M.Jackson. Combinatorial Enumeration.New York: Wiley, 1983.
[9] Christian Collberg, Ginger Myles, Michael Stepp.An empirical study of Java bytecode programs[J]. Published online 24 October 2006 in Wiley InterScience. Softw. Pract. Exper. 2007, vol. 37, pp. 581–641.
[10] Y.He.Tamperproofing a software watermark by encoding constants,Master's thesis,Comp.Sci.Dept.,Univ.of Auckland,2002.
[11] J.Nagra.Threading software watermarks.Ph.D. dissertation. University of Auckland,Auckland, New Zealand. 2006.
[12] Collberg,E.Carter,S.Debray,H.Huntwork,J.Kececioglu,C. Linn,M.Stepp. Dynamic path-based software watermarking.In: Pugh W, Chambers C,eds.Proc.of the ACM SIGPLAN 2004 Conf.on Programming Language Design and Implementation 2004,Vol 39.Washington:ACM,2004.pp.107−118.

ZHU 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.