第32卷 第20期 计 算 机 工 程 2006年10月
Vol.32 № 20 Computer Engineering October 2006
·安全技术· 文章编号：1000—3428(2006)20—0160—03 文献标识码：A 中图分类号：TP393.08

基于多常量编码的动态图软件水印保护技术

沈静博，房鼎益，吴晓南，陈晓江
(西北大学计算机科学系，西安 710127)

摘 要：提出了一种利用多常量编码伪水印来对动态图水印进行保护的方法，设计了针对IPPCT结构动态图水印的保护算法。该算法通过创建多个IPPCT结构的伪水印对宿主程序功能性的依赖关系，对真实水印起到了防篡改的作用，增加了攻击者的攻击难度，从而可对该类水印进行有效的保护。

关键词：动态图软件水印；鲁棒性；常量编码；伪水印；防篡改

Protection of Dynamic Graph Watermark Based on Multiple Constants Encoding

SHEN Jingbo, FANG Dingyi, WU Xiaonan, CHEN Xiaojiang
(Department of Computer Science, Northwestern University, Xi'an 710127)

【Abstract】 A novel approach of dynamic graph software watermark protection with multiple constant encoding is proposed, and an algorithm to protect the watermark based on improved PPCT structure is designed and implemented. Through creating multiple dependencies of fake watermarks with the protected program, the algorithm presents many significant features, such as, tamper proofing, anti-attack, and effectiveness in watermark protection.

【Key words】 Dynamic graph watermark; Robustness; Constant encoding; Fake watermark; Tamper proofing

随着Internet的不断发展和广泛应用，软件盗版问题日益严重，因此，软件保护技术已越来越被重视。软件水印(Software Watermarking)技术是目前常用的软件保护技术之一。通常可以将水印信息直接隐藏到软件程序的可执行代码或固定的数据区中，但这种静态水印很容易被攻击者识别或进行混淆攻击。Collberg和Thomborson提出了一种动态图水印(Dynamic Graph Watermark，DGW)技术[1]，它是在软件运行时动态地生成水印，并将水印信息转化成某种图结构并隐藏在软件代码中，由于这种图结构包含许多指针并且是在运行时动态生成，而指针的具体值在每次运行时都是不同的，因此相对于静态水印来说，给攻击者带来的攻击难度更大。然而，这种类型水印信息本身与宿主程序的功能性之间并无太大的关联，攻击者仍可通过深入分析，从而对该类水印进行攻击。一旦图水印结构遭到破坏就将导致水印提取过程失败。因此，研究针对这种动态图水印的有效保护措施是非常有必要的。

软件水印保护的目标是采取一切可能的手段加大攻击者的攻击难度，使其更难于分析水印程序，以防篡改。对于动态图水印的保护，本文借鉴了软件保护技术中的防篡改(Tamper proofing)思想[2]，引入了一种有效的多常量编码伪水印的方法，据此可以创建一个特殊的动态图水印系统。在这种水印系统中，真实水印和伪水印具有极相似的拓扑结构，并且由于伪水印的特殊的编码方式，一旦伪水印结构遭到攻击者破坏，将导致软件进入故障模式，致使攻击失败，因此从一定程度上增强了动态图水印的鲁棒性。

1 动态图水印(DGW)

根据数论中的大数分解难的问题，可选择一个代表版权信息的大自然数N作为水印数，并将此大数N转化成某种图拓扑结构后嵌入到软件程序代码里，由于此大数能够分解为两个足够大素数P和Q的乘积，只有合法用户才能检测到N并将其分解为P和Q，因此可证明该用户的合法身份。

1.1 DGW的嵌入与提取

DGW的嵌入是指在运行时将一个代表版权信息的大数表示成图拓扑结构并嵌入到宿主程序里，其过程具体描述为：

(1)选取一个合适的水印数N；

(2)将N表示成某种图拓扑结构；

(3)创建在运行时生成图结构的水印代码；

(4)将此水印代码嵌入到宿主程序中。

由于水印图结构的某些特性对于水印嵌入者是可知的，因此DGW的提取可通过分析软件运行期的程序内存堆栈结构来实现。

1.2 针对DGW的攻击

攻击者可以通过逆向工程、源代码/字节码分析、程序输出分析、内存堆栈分析等方法对软件水印进行破坏。通常的静态水印极易受到保持语义的代码混淆攻击，动态图水印虽然不会受到此类攻击的影响，但它也有可能受到一些更高级的攻击，例如，构成DGW结构的节点被攻击者进行了删除、拆分等篡改。

对DGW的任何一类成功的攻击都将使动态图软件水印技术失去价值。因此，有必要采取措施加强DGW的抗攻击性，其意义不仅在于对DGW的保护，更在于对软件本身的

作者简介：沈静博(1979－)，男，硕士生，主研方向：信息安全，软件保护技术，分布式系统；房鼎益，教授、博导；吴晓南，博士生；陈晓江，讲师、博士生

收稿日期：2006-01-03 **E-mail：**shen-jingbo2005@163.com

保护，从而加大了软件防盗版与防破坏的强度。

2 多常量编码的动态图伪水印方法

为了抵御对动态图水印的各种攻击，笔者提出一种多常量编码伪水印的方法，根据此方法创建了一个特殊的动态图水印系统。其基本思想是：从宿主程序中选择若干合适的常量，并将其转化成与水印图结构相似的拓扑结构，从而形成若干个所谓的“伪水印”来迷惑攻击者。除非攻击者从若干个图结构中精确地选中真实的水印图结构进行攻击，否则对任何一个伪水印图结构进行的篡改都将导致宿主程序不能正常执行。该方法实现的流程如图 1。

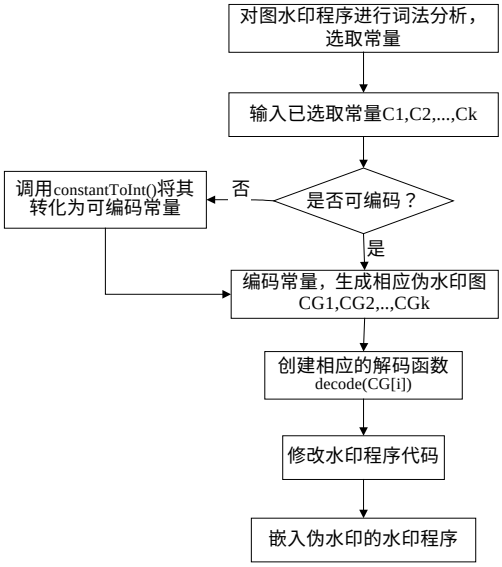

图 1 多常量编码的动态图伪水印方法实现流程

2.1 图拓扑结构的选取

图拓扑结构的选取，亦即水印数 N 的表示方法的选取，对于动态图水印系统的优劣来说是至关重要的。Collberg 和 Thomborson 在文献[1]中提出了一种 PPCT(Planted Plane Cubic Tree)结构的大数表示方法：

(1)有一个生成节点(Origin)，其唯一子节点为一个二叉树的根节点。

(2)每个节点具有左、右两个指针，非叶节点的左右指针分别指向自己的左右子节点；叶结点的右指针指向自己，左指针的指向遵从下述规则：某非叶节点的左子树的右下节点的左指针指向其右子树的左下节点。整个树的左下节点的左指针指向生成节点，生成节点的左指针指向这个树的右下节点。

(3)一个具有 m 个叶节点的 PPCT 结构有 $\frac{1}{m}\times C_{2m-2}^{m-1}$ 种表示方法。

由图 2 左部中给出的一种具有 4 个叶节点的PPCT结构可以看出，所有叶节点的右指针都指向自身，因此可以对所有叶节点的右指针加以改进，使其包含一定的信息。本文的水印系统则采用一种改进的PPCT(Improved Planted Plane Cubic Tree，IPPCT)结构表示法[5]，它相对于PPCT表示法来说能用尽可能少的节点表示尽可能大的数。于是，水印数N可以用具有m个叶节点的IPPCT结构表示成以m+1 为基数的表达式：$N=\sum_{i=0}^{m-1} e_i\times(m+1)^i$，其中系数 e_i 的值可以用相应叶节点的右指针所包含的信息来表示(即从相应叶节点右指针指向的叶节点返回原叶节点需要经过的叶节点的个数为 e_i)。图 2 右部给出了一种具有 4 个叶节点的IPPCT结构，并且用此IPPCT结构表示 $N=37\times19=703=3\times5^0+0\times5^1+4\times5^2+4\times5^3$。

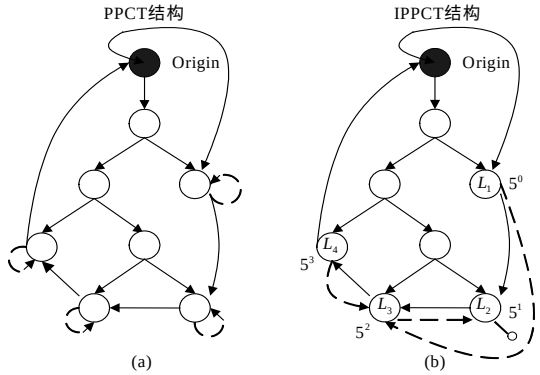

图2 一种具有4个叶节点的PPCT结构与IPPCT结构

2.2 针对 IPPCT 结构的多常量编码伪水印算法

(1)编码常量选取

首先对水印程序进行词法分析，筛选出若干合适的待编码常量放置于自定义的待编码常量列表中。对于 Java CLASS 文件，通过访问常量池操作(如：ldc, ldc_w, bipush, sipush, iconst_<i>等)可以实现对水印程序进行待编码常量的选取。

根据大数分解难问题的水印方案描述可知，转化成图结构的数应为一整型数，因此应对已选取的常量列表中非整型量进行转化，使其转化成整型。为此创建函数 constantToInt() 并重载该函数。下列关键算法描述了如何将浮点型量转化成整型量：

```
public Vector static constantToInt(float C){
//将浮点型量转化为两个整型量
int leftInt, rightInt;
leftInt = (int)C;
rightInt = (int)((C-(int)C)*10);
Vector result = new Vector();
result.add(new Integer(LeftInt));
result.add(new Integer(RightInt));
return result;
}
```

(2)常量编码

所谓常量编码就是将选中的若干待编码常量 $C_1,C_2,...,C_k$ (k 为选取的常量的个数)转化为图拓扑结构，即生成相应的伪水印图 $CG_1,CG_2,...,CG_k$。类似于动态图水印系统中水印的生成，分别创建编码函数 enCode(int i)来实现从常量到图的转换。从图 3 可以看出将常量 $C_1,C_2,...,C_k$ 进行编码后生成的伪水印图结构和水印图结构非常相似，从而可以有效地迷惑攻击者，使攻击者很难从中精确地选中水印进行攻击。

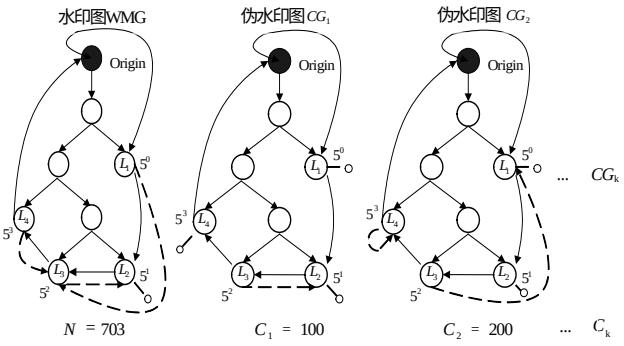

图3 真实水印与伪水印对比

(3)常量解码

常量解码是常量编码的逆过程，即将伪水印图还原为常量数。对于 IPPCT 结构，叶节点数目 m 是嵌入者已知的，系数 e_i 可以通过以下方法计算求出：

由 Origin 节点左指针指向的叶节点开始，规定 m 个叶节点分别为 L_i，其中 $i \in \{1,2,3,\ldots,m\}$，并以此代表基数的幂，那么

$$e_i = \begin{cases} 0 & L_i\text{的右指针为空} \\ j-i+1 & \text{若 } i \le j \\ m-|j-i+1| & \text{若 } i > j \end{cases}$$

其中 j 为叶节点 L_i 的右指针指向的叶节点 L_j 的下标。因此，根据公式 $N = \sum_{i=0}^{m-1} e_i \times (m+1)^i$，可以创建解码函数 deCode(IPPCT CG[i])来实现从 IPPCT 结构到常量的还原。

(4)修改水印程序代码

水印程序部分初始伪代码如下：

```
IPPCT   Wm;   //水印图为 IPPCT 结构
public method(){
    int C[1], C[2];
    …
    Wm = buildWm();   //嵌入水印
    C[1] = 100;
    C[2] = 200;
    …
    print(C[1]+ C[2]);
}
```

修改后的水印程序伪代码为：

```
IPPCT   Wm, CG[1], CG[2];   //水印与伪水印均为 IPPCT 结构
public method(){
    int C[1], C[2];
    …
    CG[1] = enCode(1); //嵌入第 1 个伪水印
    CG[2] = enCode(2); //嵌入第 1 个伪水印
    …
    Wm = buildWm();
    C[1] = deCode(CG[1]); //从第 1 个伪水印中提取常量
    C[2] = deCode(CG[2]); //从第 2 个伪水印中提取常量
    …
    print(C[1]+ C[2]);
}
```

2.3 性能讨论

根据多常量编码的伪水印方法创建的动态图水印系统中，由于水印数可以表示为 $N = \sum_{i=0}^{m-1} e_i \times (m+1)^i$，易得 $0 \le N \le (m+1)^m - 1$(其中 m 为叶节点数)。因此，系统中创建伪水印所需要的节点数如表 1 所示。

表 1 真实水印数最大值与伪水印所需节点数对应表

所能表示的真实水印数的最大值	创建伪水印所需节点数 (k 为伪水印个数)
2.6×10^{10}	20k
1.0×10^{155}	162k
2.7×10^{200}	200k

对于多常量编码的伪水印方法而言，嵌入的伪水印的数量与真实水印的抗攻击性是成正比的，也就是说嵌入的伪水印数量越多，真实水印被破坏的几率就越小，水印的抗攻击性就越强。然而，由表 1 可以看出系统在创建若干用以迷惑攻击者的伪水印的同时，势必会带来软件程序的空间过载，以至于影响软件的运行效率。因此，在抗攻击强度和软件运行效率的权衡问题上也是改进方法值得考虑的地方。

3 总结

从原理上来讲，任何保护技术都是能够被攻破的，如果能使攻击者的攻击成本大于重新开发的成本，那么这种保护技术就是成功的。本文针对 DGW 本身的保护，借鉴了防篡改技术的思想，采取了多常量编码的方法嵌入了多个与宿主程序功能性相关并且与真实水印结构相似的伪水印，尽可能地加大了攻击者的攻击成本，同时增强了真实水印的鲁棒性，因而可对 DGW 进行有效的保护。

参考文献

1 Collberg, Thomborson. Software Watermarking: Models and Dynamic Embeddings[C]. Proc. of the 26th ACM SIGPLANSIGACT, Symposium on Principles of Programming Languages, 1999.

2 Collberg, Thomborson. Watermarking, Tamper-proffing, and Obfuscation: Tools for Software Protection[J]. IEEE Transactions on Software Engineering, 2002, 28(8).

3 Palsberg J, Krishnaswamy S, Kwon M, et al. Experience with Software Watermarking[C]. Proceedings of the 16th Annual Computer Security Applications Conference, 2000: 308-316.

4 Curran D, Hurley N J, Cinnéide M Ó. Securing Java Through Software Watermarking[C]. Proceedings of the 2nd International Conference on Principles and Practice of Programming in Java, 2003.

5 王　勇，杨义先．一种基于 PPCT 的软件水印库生成方案[C]. 北京: 第三届全国信息安全会议, 2003.

(上接第 154 页)

参考文献

1 Li Shanlin, Ger-Chih C. A General Theory for Asynchronous Speech Encryption Techniques[J]. IEEE Journal on Selected Areas in Communications, 1986, 4(2): 280-287.

2 Koilpillai R D, Vaidyanathan P P. Cosine-modulated FIR Filter Banks Satisfying Perfect Reconstruction[J]. IEEE Transactions on Signal Processing, 1992, 40(4): 770-783.

3 Schafer R, Rabiner L. Design and Simulation of a Speech Analysis-synthesis System Based on Short-time Fourier Analysis[J]. IEEE Transactions on Audio and Electroacoustics, 1973, 21(3): 165-174.

4 Schafer R. A Survey of Digital Speech Processing Techniques[J]. IEEE Transactions on Audio and Electroacoustics, 1972, 20(1): 28-35.