多水印技术在软件版权管理中的应用研究

罗养霞[1,2] 房鼎益[1]

(西北大学信息科学与技术学院 西安 710069)[1] (西安财经学院信息学院 西安 710127)[2]

摘 要 为增强软件水印的鲁棒性,将图像"多水印"思想引入到软件中,并给出软件多水印定义,分析联合方式,给出模型及优化方法。在此基础上,提出一种基于多水印的软件版权保护模型,并对原有的算法进行改进,解决有意义软件水印的预处理问题、指纹的动态混淆嵌入及交互防篡改检测问题,提高软件水印的鲁棒性。实验表明,该模型在防止静态分析、动态跟踪、反逆向工程以及保护水印和软件的完整性方面具有较好的性能。

关键词 软件水印,多水印,版权管理,鲁棒性

Multi-watermarks Technology in Software Copyright Management

LUO Yang-xia[1,2] FANG Ding-yi[1]

(College of Information Science and Technology, Northwest University, Xi'an 710069, China)[1]

(Department of Information, University of Finance & Economics, Xi'an 710127, China)[2]

Abstract To enhance the robustness of software watermarking, the image "multi watermark" was introduced into the software, and gave the definition of multi-watermark of the software, analysed the joint, model and optimization method. On the basis, the multi-watermarking based on model of software copyright protection was proposed, and the original algorithm was improved, such as the meaningful pre-processing software watermarking, fingerprint embedded though the dynamic and confusion, the interactive tamper detection improves software watermark robustness. Experiments show that the model in the prevention of static analysis, dynamic tracking, and protection against reverse engineering and software integrity of the watermark has good performance.

Keywords Software watermark, Multi watermark, Software copyright management, Robustness

1 引言

软件水印是数字水印的一种,通过一定的嵌入算法,将软件的版权、用户的指纹等信息隐藏在软件中。当发生软件版权纠纷时,从软件中提取水印信息,以达到保护载体的目的。它是集密码学、软件工程、算法设计、图论、程序设计等学科的交叉研究领域。目前,水印技术在图像、音频、视频等多媒体数字产品的版权保护方面,已经是一种相对比较成熟的技术。而在软件水印方面还有待进一步发展[1],如水印的鲁棒性问题;水印的用途更多是针对版权认证,而不是用于权限限制;水印与其它技术的结合方面。研究更安全的水印算法仍是软件水印研究的重点之一。

当前,人们对水印的研究主要集中于单个水印算法,包括静态算法(基本块重排算法[2,3]、嵌入媒体水印 MC 算法[4]、应用扩频水印算法[5-7]、不透明谓词的软件水印算法[8-10]等)和动态算法(基于动态图的 CT 算法[11,12]、基于线程的 NT 算法[13]等)。这些水印算法在正常使用时,大多数情况下对软件能够起到保护作用,但是在无意或者恶意攻击下,对程序进行静态分析和动态跟踪时,单个水印很难在众多的攻击下依然不被破坏和正确检测。

到目前为止,已有的软件水印算法共约 11 种。国内外的软件水印算法多围绕这 11 种算法进行研究改进[14,15]。我们对现有水印算法在性能和抗攻击方面在 Sandmark 平台下提供了 35 个混淆算法进行扭曲攻击、11 个水印算法进行添加攻击,并对它们做参照性比较(1:重排序/重命名,2:扩展程序语义,3:改变程序统计),其结果如表 1 所列。

这些水印算法,从水印载体形式上看,有的是源码,有的是中间码,还有的是可执行文件;从程序结构上看,有的是改变了程序的控制流,有的改变了程序的指令、常量、操作码、线程等;从加载位置上看,有常量池、方法表、行号表、内存堆栈或全局变量域;从加载时刻上看,有的是静态,有的是动态;从抗攻击方法上看,对静态分析、动态跟踪、各种混淆和代码优化等攻击各有侧重,并不是面面俱到,在抵抗某一类攻击方面各有不足。然而设计出一种水印算法就可以抵抗多个方面的攻击是十分困难的。

那么人们自然会想到研究多个水印的效果。最早出现多水印的文献报道大约是在 1999 年[16],人们已经关注多水印的潜在用途。后来,吕俊贤[17]运用两种相反方式调制的水印来抵抗模糊化和直方图均衡化两种方法的联合攻击;Peterr[18]提出同时嵌入多个水印的盲水印技术,降低了图像水

到稿日期:2010-11-07 返修日期:2011-07-20 本文受陕西省教育厅项目(11JK0987),陕西省科技厅项目(2010JM8029,2011JM8016)和省科技厅项目(2009JM8007)资助。

罗养霞(1974—),女,博士生,讲师,主要研究方向为数字水印、数据挖掘;房鼎益 教授,博士生导师。

印被去除的几率。这些都是应用在以视频或图像为载体的多水印研究中，而在软件多水印方面的研究几乎为零，而且软件水印和图像水印无论在嵌入还是攻击方面差异都比较大，其方法技术并不适用于软件水印[19]。2009 年，Christian Collberg 等人在文献 [14]中针对软件攻击首次关注软件多水印问题，接下来的内容主要围绕基于多水印技术进行研究，需要研究这些水印怎样能有机地组织在一起并相互协同，以提高鲁棒性；研究应用水印和权限的绑定，以及水印和加密、防篡改等技术的结合，对软件的版权管理具有重要意义。表 1 列出软件水印算法比较。

表 1　软件水印算法比较

Algorithm Names	Code Properties	Recognition TypeCredibility	Credibility	Data Rate	Overhead		Embedding			Resistance to Attack			
					Size	Perf	1	2	3	Subt	Dist	Addi	Coll
DM	static	informed	low	low	low	low	√			weak	15	1	weak
MC	static	blind	high	high	high	low		√		weak	35	11	strong
GCB/QP	static	blind	low	low	low	low	√			weak	(28)	/	weak
SS/SHKQ/	static	informed	mediun	high	low	low			√	strong	16	0	weak
Dummy. Meth	static	blind	high	high	low	low		√		weak	22	3	weak
GTWvvs	static	blind	high	high	high	high		√		weak	29	2	/
Arboit	static	blind	high	high	low	medium		√		weak	25	2	weak
Abstract Interp	abstract	blind	high	high	/	/	√			/	/	/	/
CT	dynamic	blind	high	high	medium	/		√		feasible	34	9	weak
Path-based	dynamic	blind	high	high	low	low		√		weak	34	10	weak
ThreadBased	dynamic	informed	high	high	high	high	√			/	/	/	/

本文的研究受到多个省级项目（动态图水印及 Asmuth-Bloom 体系水印）的支持。文中给出鲁棒性、多水印的定义，研究多水印的耦合方式及其优化（第 2 节）；应用多水印对软件版权进行保护（第 3 节），给出框架模型、算法过程、权限管理和耦合方法。在原有算法的基础上提出改进，研究了水印与混淆技术、水印与防篡改技术相结合的方法，以防止对水印的静态分析、动态跟踪、逆向工程和篡改攻击。最后对此系统给出性能分析及总结展望。

2　命题定义及模型改进

为了更好地研究多水印技术增强水印的鲁棒性问题，首先对水印的鲁棒性进行清晰定义。水印的鲁棒性（Robustness）是衡量水印系统抗攻击能力的一个指标。Collberg 等人[20]对软件水印的攻击分为 4 类：添加（Additive Attacks）、去除（subtractive Attacks）、扭曲（distortive Attacks）攻击和共谋攻击（collusive Attacks），但没有对鲁棒性进行定义，只给出水印的评价标准——水印强度（Watermark Resilience），包括加载强度（Size Resilience）、空间强度（Space Resilience）和运行强度（Runtime Resilience），分别从静态水印改变程序的大小、动态水印改变程序的状态和程序的运行时间上对水印的强度进行衡量。文献[21]用其中的 γ 因子来衡量水印的鲁棒性，但是经过实验证明，一个程序的尺度或状态的变化和水印的抗攻击数并不是成正比的，也就是说强度并不能反映鲁棒性。因此，在这里仍借助鲁棒性一词来定义水印的抗攻击能力，因为在实际应用中，抵抗攻击是鲁棒性软件水印应具有的最重要的能力。

2.1　鲁棒性定义

设 T 为目前所有攻击方法的集合，$T=\{t_i\}$，$i=1,2,\cdots,N$，N 为攻击方法总数目，W_i 为某个水印算法，函数 $R_{\text{AntiAttackNum}}(P,w_i,T)$ 计算在水印算法 W_i 下软件 P 所受到的攻击集合 T 所能抵抗的最大攻击数目，或称鲁棒性。为研究方便，多水印只研究两种水印算法的联合情况，即在 w_1 和 w_2 联合加入时的鲁棒性 $R_{\text{AntiAttackNum}}(P,w_1,w_2,T)$，对鲁棒性定义如下。

定义 1（完全鲁棒）　如果 $R_{\text{AntiAttackNum}}(P,w_1,w_2,T)=N$，即在两种复合水印算法下，能抵抗 T 中所有方法的攻击，则称水印是“完全鲁棒”的。

定义 2（完全脆弱）　如果 $R_{\text{AntiAttackNum}}(P,w_1,w_2,T)=0$，即在两种复合水印算法下，不能抵抗 T 中任何方法的攻击，则称水印是“完全脆弱”的。

定义 3（强鲁棒性）　如果 $N/2<R_{\text{AntiAttackNum}}(P,w_1,w_2,T)<N$，即在两种复合水印算法下，能抵抗 T 中大部分的攻击，则称水印是“强鲁棒性”的。

定义 4（弱鲁棒性）　如果 $0<R_{\text{AntiAttackNum}}(P,w_1,w_2,T)\leqslant N/2$，即在两种复合水印算法下，不能抵抗 T 中大部分的攻击，则称水印是“弱鲁棒性”的。

以上对鲁棒性的定义同样适用于单水印算法。

2.2　多水印联合定义

为研究多水印，我们先给出相关的定义。

定义 5（单水印）　水印 w，不管是通过数字拆分，还是通过图拓扑拆分成多个部分，如 $w_1,w_2,\cdots,w_n$，如果这些水印在嵌入、提取和识别算法上都相同，而且为保护某一软件，同时被嵌入到该软件中，则称软件 p 为单水印嵌入。

定义 6（多水印）　水印 $w_1,w_2,\cdots,w_n$，如果这些水印在嵌入、提取和识别算法上都不同，而且为保护某一软件，同时被嵌入到该软件中，则称软件 p 为多水印嵌入。

即 $P(w_1\cdots w_n)\leftarrow Embed_1(P,w_1)+\cdots+Embed_n(P,w_n)$。为便于研究，我们简化多水印为两种水印进行分析，即 $P(w_1,w_2)\leftarrow Embed_1(P,w_1)+Embed_2(P,w_2)$。

对于多水印的联合，有两种方式：耦合联合水印和非耦合联合水印。

定义 7（耦合联合）　水印 w_1 和 w_2 复合嵌入同一程序 P 中，如果 w_1 和 w_2 中任一个水印在嵌入、识别或提取时和另一个水印的相关度 δ 不为零，即 $\delta(w_1,w_2)\neq 0$，则称 w_1 和 w_2 为耦合联合。当 $\delta(w_1,w_2)>0$ 时，为正耦合，联合水印抗攻击性增强；当 $\delta(w_1,w_2)<0$ 时，为负耦合，联合水印抗攻击性减弱。

例如，动态图水印 CT 算法和基于扩频水印算法 SS/SHKQ，前者是将水印通过分割转换成图拓扑，再进行图编码，定位后插入到程序中；后者是统计程序的单个指令或指令组出

现的频率，再把这些频率通过扩频变换频率而加入水印，生成特征，提取时和原程序的特征进行相似性对比，那么前者嵌入水印后，增加了程序段，改变了指令，也就改变了程序的指令频率，则肯定会影响后者频率的正确统计，产生负耦合效果。

定义 8（非耦合联合） 水印 w_1 和 w_2 复合嵌入同一程序 p 中，如果 w_1 和 w_2 中任一个水印在嵌入、识别或提取时和另一个水印的相关度 δ 为零，即 $\delta(w_1,w_2)=0$，则称 w_1 和 w_2 为非耦合联合。

例如，给程序中嵌入媒体水印的 MC 算法和与改变寄存器分配的 QP 算法，前者是给程序中插入一张图片，为了防止被去除或篡改，可调用程序中的一个函数插入到图像中；后者是通过构建指定函数的控制流图 G，再分析 G，构建干涉图 I，通过水印 W，给 I 中增加伪干涉图，以改变寄存器的分配。这两种水印在嵌入、提取和识别时，只要选择的函数不同，就没有耦合关联。

2.3 多水印数学模型及改进

对于软件同时加入两种或多种水印算法时，如联合水印 $\{w_1,w_2\}$ 对软件的抗攻击性的改变，存在几种可能：

1）当 $\delta(w_1,w_2)=0$，即为非耦合联合时，主要存在：a）强 $R(w_1)$+强 $R(w_2)$=强 $R(w_1,w_2)$；b）强 $R(w_1)$+弱 $R(w_2)$=强 $R(w_1,w_2)$；c）弱 $R(w_1)$+弱 $R(w_2)$=弱 $R(w_1,w_2)$。这 3 种情况是并列关系。对于非耦合联合，任意多个水印的鲁棒性是线性迭加关系。本身不同的嵌入方法就具有隐蔽性，所以对于非耦合水印，可通过互补迭加方式增强软件的鲁棒性。例如把一个水印（一个大数）做预处理后，分成若干个小数，要分散嵌入到软件中。目前的研究中，只是用同一种嵌入算法，把多个水印小数嵌入到软件。那么当一个水印算法被攻击，其它水印也容易被攻击。如使用多水印算法，可以将这些小数选择不同的水印算法，交错或随机使用，进行嵌入，从而增加了攻击者的难度，此模型需进一步实验验证。

2）当 $\delta(w_1,w_2)\neq 0$，即为耦合联合时，主要存在：a）强 $R(w_1)$+强 $R(w_2)$+$\delta(w_1,w_2)$=完全 $R(w_1,w_2)$（$\delta(w_1,w_2)>0$）；b）强 $R(w_1)$+强 $R(w_2)$+$\delta(w_1,w_2)$=弱 $R(w_1,w_2)$（$\delta(w_1,w_2)<0$）。这两种模型表明存在增强两种强鲁棒水印的正耦合和削弱两种强鲁棒水印的负耦合。其它弱鲁棒水印的耦合联合模型类似，这里不再赘述。可以看出，鲁棒性是增强还是减弱，主要取决于耦合方法 δ。

改进方法：基于嵌入域的无关性进行水印联合，避免负耦合，如嵌入到不同的函数中、嵌入到不同的类中、嵌入到不同形式的软体载体中。基于水印编码正相关度进行水印联合，例如以完整性验证为目标，将两个无耦合联合的水印建立联系，进行交互检验，来保护水印的完整性。则当任何一个水印被破坏，都会被检测出，进而影响整个软件的正常运行。其应用过程见第 3 节。

3 应用多水印对软件的保护

3.1 多水印软件管理模型

将抗静态分析和动态跟踪的动态图拓扑水印[22]与抗逆向工程的、类似混淆操作码水印[23]两种水印算法联合，提出多水印软件管理模型（Multi-watermarks Software Right Management Module，MWSRM）。我们以动态图水印进行开发商的版权保护，记为 W_A，通过混沌散列编码方式加入到程序中除核心代码外的各个部分；以混淆操作码水印算法进行客户指纹（代表客户信息的身份标识杂凑值）保护，记为 W_B，加入到程序核心代码段中，使客户指纹与权限绑定，并对其进行验证。为了保证软件的完整性，我们应用 DRM 播放器中的防篡改思想——交互检测算法，对两种水印进行检测和响应。如果发现水印被篡改，可自行恢复或破坏核心代码段。为了提高系统性能，我们只在核心代码和动态图的根节点进行检验。多水印软件嵌入框架如图 1 所示。

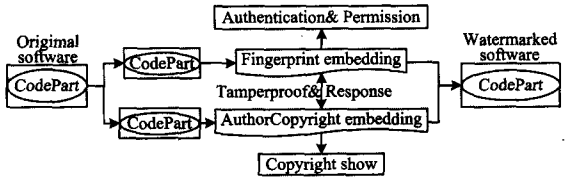

图 1 基于多水印的嵌入框架

版权水印 W_A 采用动态图拓扑算法[22]，如图 2（a）所示；客户指纹水印采用混淆操作码编码算法，如图 2（b）所示。这两种水印、两种编码方式，一个保护开发者信息，一个记录使用者信息；一个散列到程序中，一个嵌到程序核心代码区；它们保护的侧重点不同，W_A 抗静态分析和动态跟踪，W_B 抵抗逆向工程。但通过防篡改保护，改变任何一方，都会影响软件的正常使用。下面主要介绍对原有水印算法的改进部分（图中标灰色部分）。

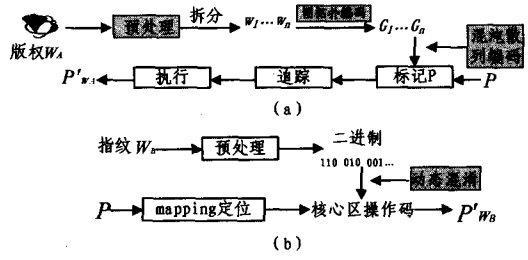

图 2 版权（a）和指纹（b）的嵌入过程

1）有意义水印的预处理

目前嵌入的软件水印多是无意义水印。有的水印还被限制在 4 个子节以内[24-26]。然而，如何嵌入一个有意义的水印，使版权更有说服力，从而增强水印的鲁棒性呢？此过程应为可逆的，水印信息 W 被预处理 D 变成可嵌入的信息 M，$M=D(W)$，其逆过程记为 $W=D^{-1}(M)$。我们主要对 Logo 图片（约小于 20kB）、身份证号\名称进行研究。通过预处理，把一个含有实际意义的水印变得可以加入到软件系统中，实现了一个独立的预处理模块。将灰度图像离散化为数字，转换为二进制序列。具体做法就是模拟灰度图像离散化为数字图像。数字图像由一个个像素组成，例如一个 512 * 512 大小的数字灰度图像就包含 512 * 512 个像素，每个像素根据亮度大小可由 8 位二进制数表示，即取值范围为 0～255。同样，对于指纹，我们选用身份证号或字母拼音名称，例如身份证号，先对 18 位数字分组，从左到右，每 4 个数为一组，分成 5 个大数；对字母拼音名称，如可以编码为 $Luo_{26}=16\times 26^2+20\times 26^1+14\times 26^0=11350_{10}$，最后应用部分和拆分算法（partial sums-splitting）把大数变成多个小数。原因有以下几点：1）机器字长的限制；2）编码字节率限制；3）小数比大数更具有隐蔽性。

把一个大数值 V 拆分成一个小数的集合 $\overline{w}=\{w_0,w_2,$

$\cdots, w_{k-1}\}$，部分和拆分算法如下：

a)计算最小指数 l，使得 V 能够被表示成 $k-1$ 个 2^l 的数。我们用 $\overline{w}$ 中的一个数 w_0 去确定 l 的值，再通过 l 计算出 $k-1$ 个拆分值。

b)把 V 分成 $k-1$ 个数 $\overline{V}=\{v_0, v_2, \cdots, v_{k-2}\}$，使得 $0 \leqslant v_j < 2^l$ 且 $V=\sum_{j=0}^{k-2} 2^{jl} v_j$；

c)另 $w_0=l-1$，则 $w_i=w_{i-1}+v_{i-1}$，得到 k 个水印数 $\overline{w}=\{w_0, w_2, \cdots, w_{k-1}\}$。

水印提取时，其逆过程如下：

a)让 $k=|\overline{w}|$，并排序，使 $w_0 \leqslant w_1 \leqslant \cdots \leqslant w_{k-1}$；

b)使 $l=w_0+1$，则当 $0 \leqslant j < k-2$ 时，有 $v_j=w_j+1-w_j$；

c)$V=\sum_{j=0}^{k-2} 2^{jl} v_j$。

例如，应用以上算法可以将一个大数 $V=1024$ 分为 5 个小数 $\overline{W}=\{4,2,2,2,2\}$，或将一个大数 $V=31415926$ 再分成 $k=10$ 个小数 $\overline{W}=\{2,8,14,15,22,27,33,40,46,47\}$，再把小数隐蔽到软件中。

2) 改进的图拓扑

程序中的图比较常见，如控制流图(CFG)、堆图(heap graphs)、注册表干涉图(register interference graphs)，有很多算法利用图拓扑编码来嵌入水印，如 CT，GTWvvs 算法。所进行的图编码方式有 OPPT(Oriented Parent-Pointer Tree)、基数图(Radix Graphs)、置换图(Permutation Graphs)、PPCT(Planted Plane Cubic Trees)、RPGs(Reducible Permutation Graphs)[27,28]。为了防止对图的 edge-flip 攻击，提高数据率，使嵌入有意义的水印成为可能，我们对 PPCT 进行改进。

PPCT 结构所有叶节点的右指针都指向自身，因此可以对叶节点的右指针进行改进，使其包含一定的信息。将改进后的 PPCT 结构称为 MPPCT 结构。MPPCT 结构保留了原 PPCT 结构的部分特征，例如具有 n 个叶节点的 MPPCT 结构、具有 $2n$ 个节点。同时，左指针满足下述规则：非叶节点的右子树的左下叶节点的左指针指向其左子树的右下叶节点，最左叶节点的左指针指向根节点，根节点的左指针指向最右下叶节点，右指针指向二叉树的根节点。

MPPCT 结构有一个 Origin 节点，即它的根节点，Origin 节点的左指针指向最右叶节点，右指针指向二叉树的根节点。每个叶节点都有两个指针：左指针指向前一叶节点，右指针满足下述原则：对于某一整数，计算它的二进制数据中连续 1 和 0 的长度。修改叶节点的右指针，从最右叶节点开始，叶节点的右指针分别指向序号为这些长度的节点，通过这种方式来表示该二进制数据。图 3 (b)给出了具有 4 个叶节点的 MPPCT 结构，此结构表示的数字 $N=1550_{10}=11000001110_2$。

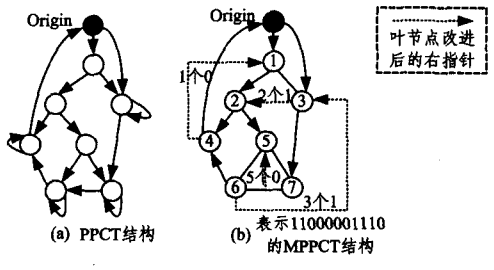

图 3 具有 4 个叶节点的 PPCT 和 MPPCT 结构对照

MPPCT 结构，既保留了 PPCT 对抗 edge-flip 攻击性，又扩展了图结构，使给定叶节点数目的 MPPCT 可以有多种结构。可以从这些结构中任意选择一种作为水印的载体，这样就形成一个较大的水印库，便于有意义水印的嵌入。

3) 混沌散列编码

为了增加水印的隐蔽性，利用混沌散列编码算法 φ 把水印信息散列在除核心代码外的各个部分。混沌替换 $C=\theta(A, B, T)$，其中 A, B 为 8-bit 的整数，T 是数字化混沌系统。用 B 替换 A 得到 C。θ 运算可以看作是以 B 为密钥，通过 T 加密 A，得到 C 的过程。其逆过程是 $A=\theta^{-1}(C, B, T)$。$S=\{s_0, s_2, \cdots, s_n\}$ 为混沌序列 $s_j \in [a, b]$，$(j=1,2,\cdots n)$，用该序列把 W 散列编码到代码 p 中，得到 $[P', R']=\varphi(w, p, S)$，其中 P' 为新代码，含有 w 的信息；R' 为保存代码，用来恢复原始代码。其逆过程是用 R' 和 P' 恢复 W 和 p，记为 $[p, w]=\varphi^{-1}(R', P', S)$。

设 w 的长度为 n 字节，P 的长度为 m 字节，$W=\{w_1, w_2, \cdots, w_n\}$，$P=\{p_1, p_2, \cdots, p_m\}$，$P'=\{p_1', p_2', \cdots, p_m'\}$，$R'=\{r_1', r_2', \cdots, r_n'\}$，$\varphi$ 算法如下：

①令 $M=m, N=n, l=\lfloor M/N \rfloor, j=1, k=0$，置 $p'=p$；

②$y=\left\lfloor l \times \frac{s_j-a}{b-a} \right\rfloor+1, k=k+y, r_j'=\theta(p_k, w_j, T), p_k'=w_j$；

③如果 $j=n$，算法结束，否则 $j=j+1$，跳到④；

④$M=M-1, N=N-1$；

⑤$l=\lfloor M/N \rfloor$，跳到②；

⑥算法结束后，$R'=\{r_1', r_2', \cdots, r_n'\}$，$P'=\{p_1', p_2', \cdots, p_m'\}$。

可逆过程 $[p, w]=\varphi^{-1}(R', P', S)$ 步骤如下：

①令 $M=m, N=n, l=\lfloor M/N \rfloor, j=1, k=0$，置 $p=p'$；

②$y=\left\lfloor l \times \frac{s_j-a}{b-a} \right\rfloor+1, k=k+y, p_k=\theta^{-1}(r_j', p_k', T), p_k'=w_j$；

③如果 $j=n$，算法结束，否则 $j=j+1$，跳到④；

④$M=M-1, N=N-1$；

⑤$l=\lfloor M/N \rfloor$，跳到②；

⑥算法结束后 $W=\{w_1, w_2, \cdots, w_n\}$，$P=\{p_1, p_2, \cdots, p_m\}$。

4) 动态混淆算法

动态混淆算法是为了把指纹加入到编译后的 PE 文件格式中，并保证水印鲁棒性和抗逆向工程。在 PE 文件中嵌入水印的方法可分为两种：静态嵌入和动态嵌入。静态嵌入：如把水印嵌入到 PE 文件的头部[29]、尾部[30]或在 PE 文件的资源结构中[31]，这些水印静态嵌入方法容易受到去除攻击；或借助静态混淆方法进行嵌入[32]，如用固定的码本对照进行操作码的混淆改变，对照的规则就是水印序列片段和操作码的一一对应关系，如图 4 所示，通过操作码的输出而解析出水印值。在编译过程中，这种一一对应关系是不改变的，多跟踪几个版本，就可以找到这种对应关系，容易受到共谋攻击。为了克服这种不足，我们利用基于指纹的动态混淆规则来嵌入水印。

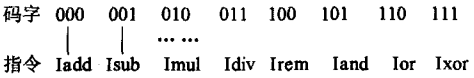

图 4 固定码本对应关系

动态混淆规则通过复合映射来实现。设有两个映射 g: $X\to Y_1$, f:$Y_2\to Z$,其中 Y_1,Y_2 则由映射 g 和 f 可以定出一个从 X 到 Z 的对应法则,它将每个 $x\in X$ 映成 $f[g(x)]\in Z$。显然,这个对应法则确定了一个从 X 到 Z 的映射,这个映射称为映射 g 和 f 构成的复合映射,记作 $f\cdot g$,即 $f\cdot g$:$X\to Z$, $f\cdot g(x)=f[g(x)]$, $x\in X$。这里 X 为指纹值,输出为指令对应关系值。举例如下。

如果要嵌入的指纹是 00011011011010110011…这样一个序列,通过 $g(x)$得到一个状态数 Y_1,如求余方式,Y_1 是 f 的一个输入,则 $f(Y_1)$的值就决定了指令与码字之间的对应状态改变值,指纹不同时,指令与码字之间的对应关系也不同。如第一个软件版本,检测时只要遇到 iadd 就译码输出 000,遇到 isub 就输出 001。但若换一个软件版本,就不是这样的对应关系。假如当某一指纹 x 经 $g(x)$计算后 $Y_1=2$, $f=g(x)+1=3$,则可能遇到 iadd 就译成 010,遇到 isub 就输出 011。这样动态的好处是防止对多个版本跟踪,找出规则进行共谋攻击。同时,混淆操作码,可以防止抗逆向攻击。

3.2 基于多水印的权限管理

大多数公司管理软件的重点不只是某个软件被盗后去应用水印诉诸法律上的认证或处罚,更多的是强调软件的防盗和限制使用,如限制软件补丁修复、功能扩展、在线升级等需求,目的是让软件使用者不能再非法深入使用、进行功能更新,或对更有价值的内容需要支付,尤其是对工业软件如组态软件、金融软件、游戏软件、财务软件的管理,如图 5 所示。客户使用一个股票分析软件试用版,当他需要访问智能分析或核心功能时没有权限,这时需要请求服务端进行升级或扩展;服务端可以通过提取客户信息进行身份认证后,发放权限。保护软件既要依靠法律的手段,又要依靠技术的手段。基于水印对软件权限的保护比加密狗等其它技术更具有隐蔽性,比加密更多一层保证,能够提供法律证明。

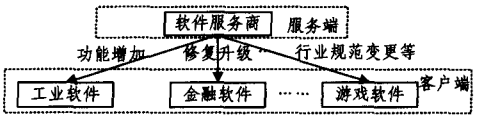

图 5 软件管理需求模式

基于多水印的权限管理分为两部分:本地版权管理和远程认证管理。第一次加载的水印数据是作品标识和版权发行机构身份标识,用来进行版权认证、有限拷贝和限制使用。第二次加载水印数据是购买者身份标识的杂凑值,用来进行身份验证,从而对软件进行远程升级或扩展管理,如图 6 所示。这里主要研究水印和权限的绑定,研究通过水印进行身份验证、放权限,没有对版权管理所涉及的其它技术进行讨论。

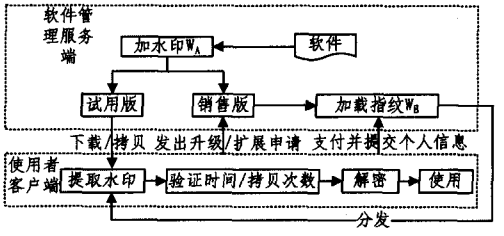

图 6 基于水印的软件权限管理

软件发行机构在自己的网页上发布软件的简介、试用版及可以提供的有偿服务信息,使用者可以通过网络下载试用版和网上交易获得销售版。试用版软件功能有限,且限制使用时间和拷贝次数。销售版软件需要购买者支付并加入其指纹信息。软件水印的加载、交易、权限验证过程如下:

Step1 开发者对软件在源码阶段进行版权水印 W_A 的加载,即 $P_{W_A}\leftarrow Embed(P,W_A,K)$, K 为水印算法密钥;

Step2 将 P_{W_A} 与软件限定使用时间 T 及拷贝次数 N 签名加密后传给版权发行机构 CP,即 CP:$Sig_k(P_{W_A},T,N)$;

Step3 使用者对试用版软件申请下载使用,运行软件时对使用时间 T 及拷贝次数 N 验证,若满足要求则继续使用,否则核心代码被破坏,软件无法正常运行;

Step4 若使用者想进行核心功能的使用或升级,可向发行机构 CP 提出申请,提供信息并完成支付,软件发行机构 CP 将申请内容传给软件版权认证中心 CA,即 $CP\to CA$;

Step5 软件版权认证中心 CA 验证正确后,将购买者身份的杂凑值预处理成指纹 WB,其信息包括购买者的身份证号、支付账号及名称,即 $W_B\leftarrow ID_1+ID_2+ID_3$;

Step6 软件版权认证中心用指纹 W_B 加在以 PE 为格式的软件 P_{W_A} 中,即 $P_{W_{AB}}\leftarrow Embed(P_{W_A},W_B,K)$;

Step7 将 $P_{W_{AB}}$ 与软件限定使用时间 T 及拷贝次数 N 签名加密后传给版权发行机构 CP,即 CP:$Sig_k(P_{W_{AB}},T,N)$;

Step8 购买者对销售版软件下载使用,运行软件时对使用时间 T 及拷贝次数 N 验证,若满足要求则继续使用,否则软件无法正常运行。

3.3 多水印耦合关联

以完整性验证为目标,将两个无耦合联合的水印建立联系,进行交互防篡改检验(见图 7)。只要有一个水印无法检测出来,就说明此软件被修改,无法正常运行,以此来保护软件版权的完整性。在这里,我们应用 DRM 播放器中的防篡改思想——交互检测算法[33],对两种水印进行检测和响应。如果发现水印被篡改,可自行恢复或破坏核心代码段。为了提高系统性能,我们只在核心代码和动态图的根节点进行验证。具体算法如下:

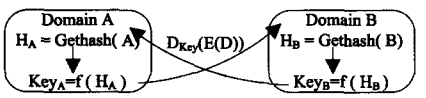

图 7 两水印耦合交互检测

1)在运行代码模块之前,程序控制器跳到指纹核心模块 *Domain B* 处,计算哈希值 $H_B=Gethash(B)$;

2)由哈希值得到解密密钥 $Key_B=f(H_B)$或 $Key_B=H_B$;

3)用 Key_B 去解密含版权头节点模块 *Domain A*, $D_{Key}(E(Domain\ A))$;

4)计算哈希值 $H_A=Gethash(A)$;

5)得到解密密钥 $Key_A=f(H_A)$或 $Key_A=H_A$;

6)用 Key_A 去解密含指纹核心模块 *Domain B*, $D_{Key}(E(Domain\ B))$;

7)解密成功,软件正常运行,否则不能正常运行。

4 实验比较

为了验证本文所提出的多水印方案的可行性,我们对原型系统——基于多水印的软件版权管理(MW-SRM)进行实

验，针对56个C++程序实现了所提出的方案，采用身份证18位数字设置水印长度，进行随机检测。实验表明，对未嵌入水印和嵌入一个或多个水印的实验对象分别用随机密钥进行检测，结果没有一例检出水印。对嵌入水印的实验对象用正确密钥检测，结果全部检出水印。对嵌入水印的实验对象多次嵌入另外水印，用正确密钥检测，结果检出所有水印。

表2 PPCT与MPPCT数据率对比表

节点数目	PPCT	MPPCT
m	$\frac{2}{m}C_{m-1}^{(m-2)/2}$	$(m-1)^{m/2}$
10	14	5.9×10^{4}
100	5.09×10^{26}	6.05×10^{99}

将PPCT和MPPCT两种水印图拓扑数据率进行比较，在节点数相同时，MPPCT能隐藏较多的信息量。表2对两者子系统在水印加载前后程序大小和运行时间差进行比较，部分数据如表3所列。不同的图拓扑水印子系统，空间差相对比较稳定，并不随宿主程序的变换而发生变化。而且在实验过程中发现，在嵌入的水印信息量相同的情况下，宿主程序越大，水印系统所带来的空间差越不明显。

表3 对PPCT与MPPCT子水印嵌入系统性能差对比
（数据值为嵌入后-嵌入前，水印数1，节点数9）

测试程序	嵌入水印前后程序大小变化差（单位：kB）		嵌入水印前后程序执行时间差（单位：kB）	
	PPCT	MPPCT	PPCT	MPPCT
Factorial	2.98	2.99	165250	166012
combination	2.97	2.97	4594	5925
Panel	3.01	3.01	2703	7594
myFrame	3.01	3.01	709	1932
valueChanged	3.02	3.02	125	140

以Sandmark为平台，用35个混淆算法和11个水印（包括算法本身）（见表4），对嵌入水印的实验对象做以下攻击：反编译再重新编译；代码混淆和代码优化；合并随机选择的类文件。攻击后分别用正确密钥检测，结果全部检测出水印。

表4 复合嵌入测试系统结果

Embed Method	Code Properties	Recongnition Type	Credibility	Data Rate	Overhead		Resistance to Attack			
					Size	Perf	Subt	Dist	Addi	Coll
CT+PE混淆嵌入	dynamic	blind	high	high	high	low	strong	34	9	strong

系统选择的是CT算法嵌入版权和PE动态混淆过程嵌入指纹，算法本身具有抗攻击性。对于单个CT算法本身，我们在同一个函数中嵌入水印。如果不做标记，或没有正确的密钥输入，则不能正确检测出水印。所以对于多水印，基于不同域嵌入效果会更好一些，如嵌入到不同的函数或类中。

结束语 基于多水印的软件保护方案是应对计算机网络环境和软件不断发展的服务模式而提出的，对软件的合法版权提出了新的保护手段。随着信息隐藏技术的发展，使得软件版权信息从简单的静态技术向动态技术发展，也使得软件水印不再局限于版权的认证，而是参与权限的约束和保护。为了提高水印的鲁棒性，将图像"多水印"思想引入到软件中，对软件多水印给出定义，对联合方式进行分析，给出模型及优化方法，并将多水印应用于软件版权的保护中，从防止水印静态分析、动态跟踪、反逆向工程、保护完整性方面来增强水印的鲁棒性。

随着研究的深入，基于多水印对软件的保护技术需要进一步研究的工作包括软件多水印加载自动化；基于水印的远程认证；如何权衡软件水印隐蔽性、数据率及鲁棒性三者之间的关系。

参考文献

[1] 张立和，杨义先，等. 软件水印综述[J]. 软件学报，2003，14(2)：268-276

[2] Collberg C, Thomborson C, Low D. Breaking abstractions and unstructuring data structures [M]. Springer, 1998:103-116

[3] Collberg C, Thomborson C, Low D. Manufacturing cheap, resilient, and stealthy opaque constructs [A]// Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages[C]. 1998:184-196

[4] Moskowitz S, Cooperman M. Method for stega-cipher protection of computer code[P]. United States Patent:5745569, 1998

[5] Stern J, Hachez G, Koeune F, et al. Robust object watermarking: Application to code[A]// Lecture Notes in Computer Science[C]. 2000:368-378

[6] Sahoo T, Collberg C. Software watermarking in the frequency domain: Implementation, analysis, and attacks[R]. TR04-07. Computer Science Department, University of Arizona (USA), 2004

[7] 吴建军，高济. 基于指令统计分布的软件水印方案[J]. 浙江大学学报：工学版，2005，39(2)：225-228

[8] Lida M H, Matsumoto K, et al. A practical method for watermarking java programs[A]// The 24th Computer Software and Applications Conference (compsac2000)[C]. 2000:191-197

[9] Arboit G. A method for watermarking java programs via opaque predicates[A]// The Fifth International Conference on Electronic Commerce Research (ICECR-5)[C]. 2002:102-110

[10] Fukushima K, Sakurai K. A software fingerprinting scheme for java using classfiles obfuscation[A]// LNCS 2908[C]. 2003:303-316

[11] Collberg C, Thomborson C, Townsend G. Dynamic graph-based software watermarking[R]. TR04-08. Univ. of Arizona, 2004

[12] Venkatesan R, Vazirani V, Sinha S. A graph theoretic approach to software watermarking[A]// Lecture Notes in Computer Science[C]. 2001:157-168

[13] Nagra J, Thomborsonm C. Threading software watermarks [M]. Berlin Heidelberg: Springer-Verlag, 2004:208-223

[14] Collberg C, Nagra J. Surreptitious Software Obfuscation, Watermarking, and Tamperproofing for Software Protection[M]. Addison-Wesley, July 2009:539-540

[15] Myles G M. Software theft detection through program identification[D]. University of Arizona, 2006

[16] Mintzer F, Braudaway G W. If one watermark is good, are more better? [C]// Proceedings of the International Conference on Acoustics, Speech, and Signal Processing. Phoenix, AZ, USA, 1999:2067-2070

[17] Lu Chun-shien, Huang S, Sze Ch, et al. Cocktail watermarking for digital image protection[J]. IEEE Transactions on Multimedia, 2000, 2(4):209-224

[18] Wong P H W, Au O C. A novel blind multiple watermarking technique for images[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, 13(8): 813-830

[19] 陈真勇，唐龙，等. 以鲁棒为目标的数字多水印研究[J]. 计算机学报，2006，29(11)：2037-2043

[20] Collberg C, Thomborson J. Software watermarking: models and dynamic embeddings[C]// Aiken A, et al., eds. Proceedings of the 26th Annual SIGPLAN-AIGACT Symposium on Principles of Programming Languages (POPL'99). Association for Computing Machinery Press, 1999: 311-324

[21] 沈静博，房鼎益，等. 基于多常量编码的动态图软件水印保护技术[J]. 计算机工程，2006，32(10)：160-162

[22] Luo Yang-xia, Fang Ding-yi. Dynamic Graph Watermark Algorithm Based on the Threshold Scheme[C]// International Symposium on Computer Science and Computational Technology (ISCSCT2008). 2008

[23] 陈亮. 基于代码混淆的软件水印方案的设计与实现[M]. 赣州：江西理工大学，2009

[24] Sandmark3. 4[EB/OL]. http://sandmark. cs. arizona. edu

[25] DynamicJava[EB/OL]. http://koala. ilog. fr/djava, 2004

[26] JavaWiz[EB/OL]. http://www. cs. purdue. edu/homes/madi/wm/, 2004

[27] Collberg C, Huntwork A, Carter E, et al. Graph theoretic software watermarks: Implementation, analysis, and attacks[C]// Workshop on Information Hiding. Springer-Verlag. 2004: 192-207

[28] Collberg C, Thomborson C, Townsend G. Dynamic graph-based software fingerprinting[J]. TOPLAS, 2007, 29(6): 1-67

[29] 吴振强，冯绍东，等. PE 文件的信息隐藏方案与实现[J]. 计算机工程与应用，2005，27：148-163

[30] 黄强波，唐傅，等. 基于 PE 可执行文件的软件水印研究[J]. 长沙航空学院学报，2009，9(3)：76-81

[31] 徐晓静，徐向阳，等. 基于 PE 文件资源结构的水印算法[J]. 计算机工程与设计，2007，28(23)：5802-5804

[32] 孙光，孙星明，等. 可嵌入水印的混淆编译框架[J]. 科学技术与工程，2005，5(10)：1671-1815

[33] Atallah M J, Hoi C. Method and system for tamperproofing software[R]. Assigned to Purdue Research Foundation, February 2006

（上接第 86 页）

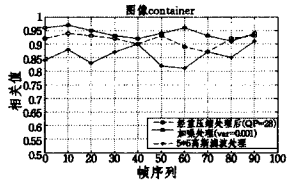

图 9 嵌入水印的 Container 视频序列受攻击情况图

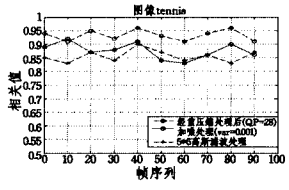

图 10 嵌入水印的 tennis 视频序列受攻击情况图

实验结果表明，水印在受到重压缩攻击时，归一化相关系数平均高于 0.93，在高斯滤波和加噪处理下也都有较好的鲁棒性，比 G. Qiu 等提出的混编视频水印方案在重压缩和其他信号处理攻击下具有更好的抗攻击能力。

4.3 水印的完整性认证能力分析

实验中分析了脆弱性水印在帧替换、帧重组、帧删除、重压缩攻击及一般信号处理下的敏感程度。如图 11、图 12 所示。

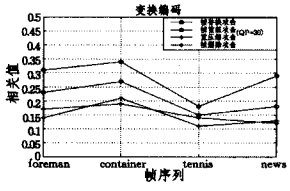

图 11 变换编码处理下水印的完整性认证情况图

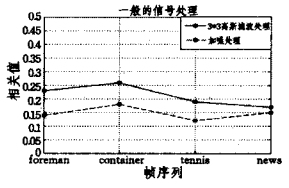

图 12 一般信号处理下水印的完整性认证情况图

由结果可知，提取的脆弱性水印与原始水印之间的相关性大部分在 0.1～0.3 之间。可见，嵌入的脆弱性水印对一般攻击和信号处理有较高敏感性，对视频的完整性认证能力较强。水印对攻击的敏感程度比 G. Qiu 等提出的混编视频水印方案更具敏感性。

结束语 由于视频的版权和完整性是视频中需要重点保护的两个对象，为实现版权保护和完整性认证的双重功能，本文提出一种具有鲁棒性和脆弱性的 H. 264 改进混编视频水印方案和盲检测技术。本方案能够在保持稳定编码率的情况下，同时嵌入鲁棒性和脆弱性水印信息，能满足实时应用的需求。实验表明，该方案具有良好的不可见性，鲁棒性水印的稳健性强，脆弱性水印的敏感性较高、认证能力较好，为进一步研究基于 H. 264 视频水印算法，提供了一种切实可行的研究思路。今后我们将继续深入研究视频水印方面的几个关键性问题，主要包括：(1)设计水印嵌入位置的选择算法；(2)保护原始水印信息的方法；(3)如何实现水印嵌入算法中拉格朗日开销的最优化；(4)运动矢量嵌入水印信息后，其方向和角度变化带来的影响。

参 考 文 献

[1] Ali M A, Edirisinghe E A. Watermarking H. 264/AVC By Modifying DC Coefficients[C]// International Conference on CyberWorlds. 2009: 241-245

[2] 汪波，冯久超. 基于 H. 264 标准的视频混沌密写算法[J]. 吉林大学学报，2008，7(4)：960-965

[3] Noorkami M, Mersereau R M. Compressed-domain Video Watermarking for H. 264[C]// IEEE International Conference on Image Processing. 2005, 9: 890-893

[4] Lu T T, Hsu W L, Chang P C. Blind Video Watermarking for H. 264[C]// Proc. IEEE Int. Conf. Electrical and Computer Engineering. May 2006: 2353-2356

[5] Qiu G, Marziliano P, Ho A T S, et al. A hybrid watermarking scheme for H. 264/AVC video[C]// Proc. 17th Int. Conf. Pattern Recognition. 2004, 4(4): 865-868

[6] 张立和，米晓莉，孔祥维. 基于运动速度的视频水印算法[J]. 计算机应用研究，2007，24(4)：187-189

[7] Noorkamia M, Mersereau R M. Digital video watermarking in P. frames with controlled video bit-rate increase[J]. IEEE Trans on Forensics and Security, 2008, 3(3): 441-455

[8] Liu Quan, Liu Hong. A novel real-time video digital watermarking algotithm[C]// Proc. of the 7th IEEE International Conference on Machine Learning and Cybemetics. 2008: 2739-2743

[9] Kuo Tien-ying, Lo Yi-chung, Lin Chen-i. fragile video waterming technique by motion field embedding with rate-distortion minimization[C]// Proc. of International Conference on Intellignt Information Hiding and Multimedia Signal Processing Washington DC: IEEE Computer Society. 2008: 853-856

[10] 孙圣和，陆哲明，等. 数字水印技术及应用[M]. 北京：科学出版社，2004：75-76