混沌不透明谓词在代码混淆中的研究与应用

苏 庆 吴伟民 李忠良 李景樑 陈为德
(广东工业大学计算机学院 广州 510006)

摘 要 通过改进Logistic混沌映射,提出了En_Logistic映射,将该映射应用到不透明谓词簇的构造过程中,形成混沌不透明谓词。将混沌不透明谓词应用于代码混淆过程。分别给出在程序分支判断点处和在顺序执行语句块中插入混沌不透明谓词的方法。对应用不透明谓词进行代码混淆的过程给出了程序复杂度评价以及控制流复杂度评价。安全性分析表明,混沌不透明谓词具备对抗动、静态攻击的安全性,并通过实验验证了其在代码混淆应用中的效果。

关键词 混沌理论,Logistic映射,En_Logistic映射,不透明谓词,代码混淆,软件保护

中图法分类号 TP309.7 **文献标识码** A

Research and Application of Chaos Opaque Predicate in Code Obfuscation

SU Qing WU Wei-min LI Zhong-liang LI Jing-liang CHEN Wei-de
(Faculty of Computer, Guangdong University of Technology, Guangzhou 510006, China)

Abstract Constructing the safety opaque predicate is the major issue in code obfuscation and the key to it lies in using opaque predicates. In order to form a chaotic opaque predicate, the En_Logistic was proposed through improving the Logistic chaotic, and then it was applied to the construction of the opaque predicate clusters. The chaotic opaque was applied in the code obfuscation process. In this experiment, different methods of inserting chaos opaque predicate into program branches and sequence blocks were given. The program complexity evaluation and the control flow complexity evaluation were given in the application process of code obfuscation. Additionally, security analysis shows that the chaotic opaque predicate has higher security in fighting against static attacks, as well as the dynamic. And better effectiveness in the code obfuscation is verified in this experiment.

Keywords Chaos theory, Logistic mapping, En_Logistic mapping, Opaque predicate, Code obfuscation, Software protection

1 引言

近年来由于逆向工程技术的迅速发展,越来越多的有价值的程序核心算法通过逆向分析窃取。可靠性是衡量软件系统的最重要特征之一[1]。为了提高软件可靠性,代码混淆作为一种抗软件逆向分析的方法随之产生。不透明谓词是一类近年出现的代码混淆新方法。它能够在保持程序的运行结果不变的前提下,通过改变程序的控制流来进行代码混淆。文献[2-4]都提及用代码混淆技术来保护软件的思想,使用一些在理论上为真的不透明谓词以条件判断表达式的形式插入至源代码中。之后,Arboit继续提出了使用二次乘余理论构造更复杂的不透明谓词的方法。袁征提出了一种利用初等数论里面的同余方程来构造不透明谓词的方法[5]。这些不透明谓词存在形式过于简单、安全性差的缺点,在逆向分析过程中较容易被过滤。

为克服利用简单方法构造不透明谓词的缺点,本文在综合各种不透明谓词构造方法的基础上,将混沌理论与不透明谓词构造方法相结合,提出了一种新的混沌不透明谓词的构造方法。该方法构造的混沌不透明谓词不仅具备较好的混淆代码性能,而且在安全性和抵抗逆向分析能力方面都较简单不透明谓词有提高。

本文将首先介绍混沌不透明谓词的构造方法,以及如何将混沌理论结合到不透明谓词簇的构造中;接着将介绍如何将不透明谓词插入到代码中;最后将对混淆的安全性进行分析。

2 混沌不透明谓词簇构造方法

不透明谓词技术首先被应用于代码混淆领域。最简单的不透明谓词的构造是利用一些数学理论上为真的数学表达式来构造的。下面给出一些与不透明谓词相关的定义。

定义1(不透明谓词) 对于一个谓词P,如果程序中点p的输出在嵌入程序之前就已知,则该谓词P是不透明的。如果谓词P的输出永远为真,则记为P^T;如果谓词P的输出永远为假,则记为P^F;如果谓词的输出有时为真有时为假,就记为$P^?$。

定义2(陷门不透明谓词[6]) 令$j\in(1,2,\cdots,n)$,K_j为

到稿日期:2012-08-09 返修日期:2012-12-02

苏 庆(1979-),男,硕士,讲师,主要研究方向为系统介入、软件测试、可视计算等,E-mail:sqsavage@163.com;吴伟民(1956-),男,教授,硕士生导师,主要研究方向为可视计算、系统工具与平台;李忠良(1985-),硕士生,主要研究方向为可视计算。

谓词 P 的密钥，若 K_j 已知，则混淆前很容易确定谓词 P 在程序中点 p 上的输出，否则若 K_j 未知，则混淆前很难确定 P 在程序中点 p 上的输出，则称谓词为陷门不透明谓词。

定义 3(不透明方法[7]) 一个返回布尔值的方法 M 如果在程序中的调用点 p 的返回值在嵌入程序时就已知，则该方法 M 就是不透明方法；如果方法 M 的返回值永远为真则记为 M^T，如果方法 M 的返回值永远是假则记为 M^F，如果方法 M 的返回值有时为真有时为假则记为 $M^?$。

2.1 映射的选择

为构造安全性高的混沌不透明谓词，相应混沌映射的选择应遵循以下原则：

1. 迭代生成的实数序列应具有较高的伪随机性。伪随机性越高，迭代生成的实数序列的统计特性就越不明显，从而增大了根据当前的状态值推测出后面的实数的难度。

2. 迭代生成的实数序列对参数和初值敏感。如果混沌映射对初始值和参数敏感，则迭代生成的实数序列将会因为初始值和参数的微小变化而产生巨大的变化。这个特性将有利于增加整个系统的不确定性。

一维 Logistic 映射是一个差分方程，如式(1)所示。

$$x_{n+1}=x_n*\mu*(1-x_n),\mu\in[0,4],x_0\in(0,1) \tag{1}$$

式中，μ 取值在越接近 4 时，迭代生成实数值越均匀分布在 0 到 1 之间。如果 μ 的取值确定，将初始值 x_0 的取值进行微小的改变，迭代生成的实数值序列将产生显著的变化，如图 1 所示。

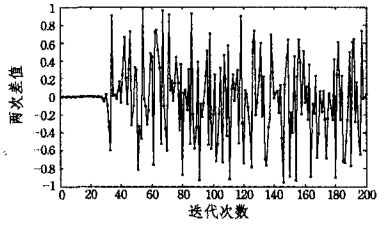

图 1 Logistic 映射在不同的初始条件下生成的实数之差

图 1 显示的是 $x_0=0.663489000$ 和 $x_0=0.663489001$，$\mu=3.99$ 时生成的两个 Logistic 实数序列值之差的图像。从图中可以看出，在最开始 30 多次迭代过程中，两者的数值之差很小。但随着迭代次数的增加，生成的实数值序列差别越来越大。

由于 Logistic 映射的吸引子在多次迭代之后趋近某一固定值 $x_n=x_{n+1}$，即在某个区间内的点容易聚集，因此在 Logistic 混沌映射的基础上加以改进，引入多个参数 λ,α,β，得到一个新的混沌映射[8]，暂命名为 En_Logistic 映射。

假设：$g(\lambda,\alpha,\beta,x_n)=\lambda x_n(\alpha-x_n)^\beta$，则 λ,α,β 应满足以下条件：

1. $\lambda>0,\alpha>0,\beta>0$。

2. 函数固定点斜率的模必须大于等于 1，此时固定点不稳定，同时函数在不可忽略固定点的梯度值越大，映射的混沌性能越好。

3. 如果不可忽略固定点的位置位于函数极大值的右侧，则必须使 $\mathrm{d}g(x')/\mathrm{d}x$ 尽可能为负值，其中 x' 为固定点，如果在左侧，则为正。

由此得出式(2)。

$$x_{n+1}=(\beta+1)(1+\frac{1}{\beta})^\beta x_n(1-x_n)^\beta \tag{2}$$
$$\beta\in(1,4),x_0\in(0,1)$$

由于 En_Logistic 混沌映射是在 Logistic 混沌映射的基础上改进而来，因此 β 取值越靠近 4 时，迭代生成的实数值会越均匀地分布在 0 到 1 之间。改进之后的混沌映射生成的实数序列的伪随机性越高，对初始值和参数越敏感。同样在 β 的取值确定之后，对初始值 x_0 做微小的改变之后生成的实数序列也将产生显著的变化，如图 2 所示。

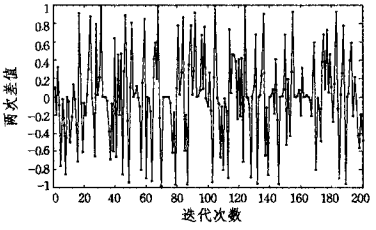

图 2 En_Logistic 映射在不同的初始条件下生成的实数之差

图 2 所示的是 $x_0=0.6,\beta=3.99$ 迭代 200 次与 $x_0=0.5,\beta=3.98$ 迭代 200 次之后生成实数值的差。可以看出结果比 Logistic 映射更好，两次生成的实数的相同率更低。

通过以上的分析看出两个混沌映射表现出较好的伪随机性和对初始值、参数的敏感性。所以本文选择这两个混沌映射来构造不透明谓词。

2.2 构造方法

不透明谓词簇的维度可以作为反映不透明谓词复杂度的度量。维度越大，则不透明谓词簇包含的不透明谓词的个数越多，其抵抗攻击的性能越强。通过控制混沌系统迭代的次数可以生成不同长度的混沌序列。令 $i,j\in(1,2,\cdots,n)$，利用混沌系统构造不透明谓词簇 $\{O_{ij}\}_{i,j=1}^{n}$，通过增加迭代的次数来增加谓词的 O_{ij} 的长度，谓词 O_{ij} 的表示为 $O_{ij}=(x_i,y_j)$。谓词的二元组表示形式中 x,y 的值将由混沌映射系统迭代生成的实数序列分别表示。

1. 将 Logistic 映射生成的实数序列作为 x_i 的取值。当参数 μ 的值越接近 4 时，生成实数序列的随机性越好。设初始参数为 μ，迭代的初始值为 x_0，则 μ,x_0 共同构成密钥 $key(\mu,x_0)$。迭代生成的实数序列的集合表示为 $X_{(\mu,x_0)}=\{x_i\mid x_{i+1}=\mu x_i(1-x_i),i=1,2,\cdots,n\}$。

2. 将 En_Logistic 混沌映射生成的实数序列作为 y_j 的取值。设初始参数为 β，迭代初始值为 y_0，则 β,y_0 共同构成密钥 $key(\beta,y_0)$。迭代生成的实数序列的集合表示为 $Y_{(\beta,y_0)}=\{y_j\mid y_{j+1}=(\beta+1)(1+1/\beta)^\beta y_j(1-y_j)^\beta,j=1,2,\cdots,n\}$。

谓词 O_{ij} 可表示为 $O_{ij}=\{(x_i,y_j)\mid x_i\in X_{(\mu,x_0)},y_j\in Y_{(\beta,y_0)},i,j=1,2,\cdots,n\}$。由于混沌系统对初值敏感，为确保系统进入混沌状态，初值 x_0,y_0 和参数 μ,β 的取值非常重要。当初值和参数都确定之后，混沌系统产生的数值序列就确定，迭代次数 i 和 j 确定之后，就可以确定相应的谓词 O_{ij}。逆向分析者如果未能获悉集合 $X_{(\mu,x_0)}$，$Y_{(\beta,y_0)}$ 的生成规则以及相

应的初值、参数和迭代的步长，是很难确定不透明谓词的，所以利用这种方式构造的不透明谓词具有极高的保密性。

参数 μ,β、初始值 x_0,y_0、迭代的步长 i,j 共同构成了该不透明谓词 O_{ij} 的密钥，用一个六元组表示为 $key=(\mu,\beta,x_0,y_0,i,j)$。$key$ 将在确定混沌不透明谓词输出时使用。

2.3 混沌不透明谓词的插入

不透明谓词的插入是混淆代码的过程的关键步骤，插入点的选择以及插入的方式都会影响混淆的效果。不透明谓词的输出是一个布尔值，所以插入点可以选择在程序中所有需要布尔值的地方，也可以利用不透明谓词的布尔输出值来构造程序分支。常用的插入点选择方式有两种：

1. 插入至需要使用布尔值进行程序分支走向控制之处。例如 if 语句的判断条件或循环语句的边界条件判断处。

2. 在顺序执行的语句块中间插入谓词，将顺序执行的控制流转换为分支控制流。谓词插入后不影响程序的执行过程，即实际的运行过程依旧与原顺序执行的过程相同。

2.3.1 在分支判断点插入谓词

在程序中的分支点处，往往需要使用布尔值确定程序控制流方向，而不透明谓词的输出也是布尔值，所以不透明谓词的常规插入点之一就是程序中的分支判断点，以及循环中的边界条件判断点等处。由于不能改变程序原始控制流方向，因此在这些地方应当使用永真谓词或永假谓词。

下面以在 if 条件下判断表达式插入为例来说明插入的方法。设原始判断条件表达式为 E_O。

1. 插入永真不透明谓词。插入 if 的条件判断式时，将谓词和 if 语句原始判断条件表达式作与运算，即插入之后的表示式为 $E_O\&\&P^T$。如果原始判断语句的输出结果为真，则和永真的不透明谓词做与运算之后结果依然为真；如果原始的判断语句的输出结果为假，则与永真的不透明谓词相与之后结果依然为假。其真值表如表 1 所列。

表 1 插入永真不透明谓词的真值表

E_O	P^T	$E_O\&\&P^T$
T	T	T
F	T	F

从表 1 中可以看出，插入之后的布尔运算结果和原始条件表达式的计算结果相同，即插入永真不透明谓词之后不会改变原来程序的计算流程。永真不透明谓词的插入方式如图 3 所示。

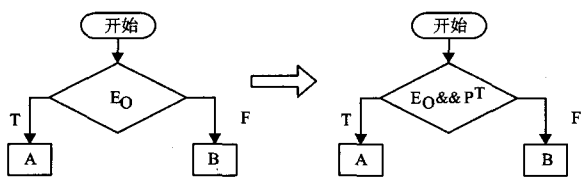

图 3 条件判断表达式中插入永真谓词

2. 插入永假不透明谓词。插入 if 的条件判断式中时，将谓词和 if 语句原始的判断条件相或，即插入之后的表示式为 $E_O \parallel P^F$。如果原始判断语句的输出结果为真，则与永假的不透明谓词相或之后结果依然为真；如果原始的判断语句的输出结果为假，则与永假的不透明谓词相或之后结果依然为假。其真值表如表 2 所列。

表 2 插入永假不透明谓词的真值表

E_O	P^F	$E_O \parallel P^F$
T	F	T
F	F	F

从表 2 中可以看出，插入之后的布尔运算结果和原始条件表达式的计算结果相同。永假不透明谓词的插入方式如图 4 所示。

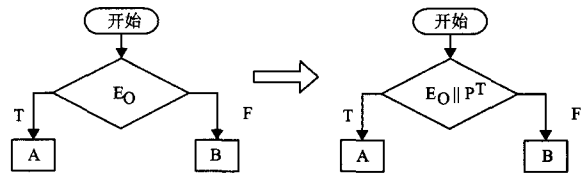

图 4 条件判断表达式中插入永假谓词

2.3.2 在顺序执行语句块中插入不透明谓词

在程序的顺序执行块中间插入不透明谓词构造分支控制流，可以对原程序的控制流作出比较大的改变。这种插入方式将本来的顺序流程变成了现在的选择分支，但是实际的效果和顺序执行一样。具体的插入方式如下：

1. 插入永真不透明谓词。插入之后原来顺序执行的部分要放在分支结构中为真的部分，为假的部分实际上永远也不会执行到，在图 5 中这部分用空方框表示。插入的方式如图 5 所示。

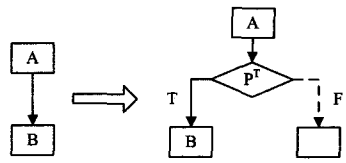

图 5 顺序执行块中插入永真谓词

2. 插入永假不透明谓词。插入之后原来顺序执行的部分要放在分支结构中为假的部分，为真的部分实际上永远也不会执行到，在图 6 中这部分用空方框表示。插入的方式如图 6 所示。

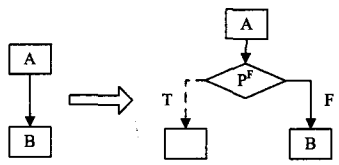

图 6 顺序执行块中插入永假谓词

3. 插入不确定不透明谓词。即不透明谓词为可满足式，输出结果有时为真有时为假。采用的插入方式是将真和假两条执行路径中执行代码的功能转变为相同，如图 7 所示。

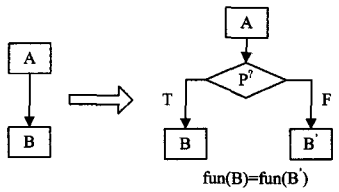

图 7 顺序执行块中插入不确定谓词

3 混淆性能评价

对混淆的性能进行评价，主要是衡量混淆转换的有效性。

对混淆前后的程序实施逆向分析，每一个子过程中二者属性的对比都可以反映混淆算法的有效性[9]。下面将对混淆转换前后程序的复杂度和控制流程图复杂度的变化进行实验对比。

3.1 混淆转换后程序的复杂度评价

混淆转换之后程序的复杂性和难以阅读分析的程度是否较转换之前更强，这项评价标准能反映混淆转换的强度。对混淆转换过程的混淆强度还没有一个具体的度量标准，但是可以借用软件工程中软件复杂度的度量方法来近似衡量。在软件工程领域，这些指标包括软件的可读性、可靠性和可维护性等，它们是在统计源代码的一些文本属性的基础上得出的，所以能在一定程度上反映出程序的复杂度。

实验中使用第三方的 Sandmark 工具[10]来统计程序的复杂度。程序复杂度的计算方法有多种，不同的计算方式从不同的方面考量程序的复杂度。本实验中使用九宫格程序作为测试程序，使用转换工具分别打包混淆前后的程序。用 Sandmark 工具分析 jar 包之后计算混淆前后的各项复杂度指标，如表 3 所列。

表 3 混淆前后复杂度对比

复杂度指标	混淆前	混淆后
数据结构复杂度(Data Structure Complexity)	31	35
扇入扇出复杂度(Fan-in Fan-out Complexity)	441	3025
圈复杂度(Cyclomatic Complexity)	17	44
类难度(Class Difficulty)	49	88
类耦合度(Class Coupling)	14	31
类功能强度(Class Effort)	43512	139480

从表 3 中的统计结果可知，混淆之后程序的各项复杂度统计指标都显著增强，各项代码属性的统计结果也比混淆之前好。

3.2 混淆转换后控制流的复杂度评价

应用不透明谓词来混淆代码着眼于对程序的控制流进行修改，所以比较混淆前后程序的控制流程图也是衡量混淆效果的一项重要方法。Sandmark 工具也提供了生成程序的控制流程图的功能模块，测试程序仍然选择九宫格程序。该模块会对字节码进行分块，并通过计算生成程序的控制流程图。图 8 是混淆前的原始程序控制流程图。

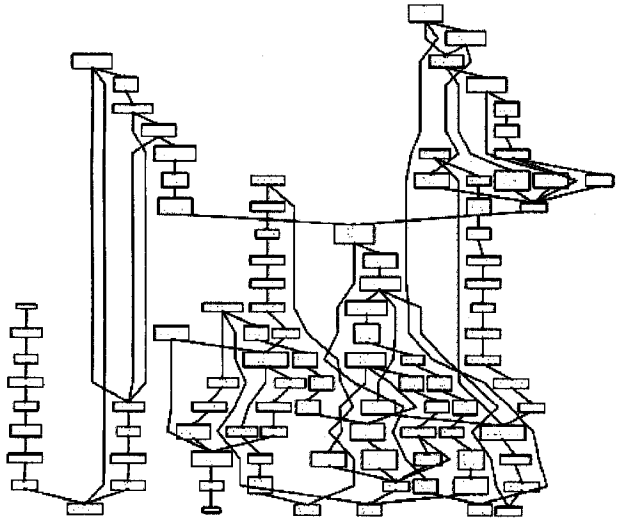

图 8 混淆前九宫格程序的 CFG

加入不透明谓词混淆后，程序的控制流程图发生了明显的变化。程序真正的执行路径已被隐藏在复杂的控制流程中，体现为程序控制流程图变得更加复杂，如图 9 所示。复杂化后的所有程序执行路径中，有些是不可能被到执行的，但是它们的存在增加了利用控制流程图分析的难度，因此逆向工程人员用程序控制流程图分析的工作量将会大大增加。

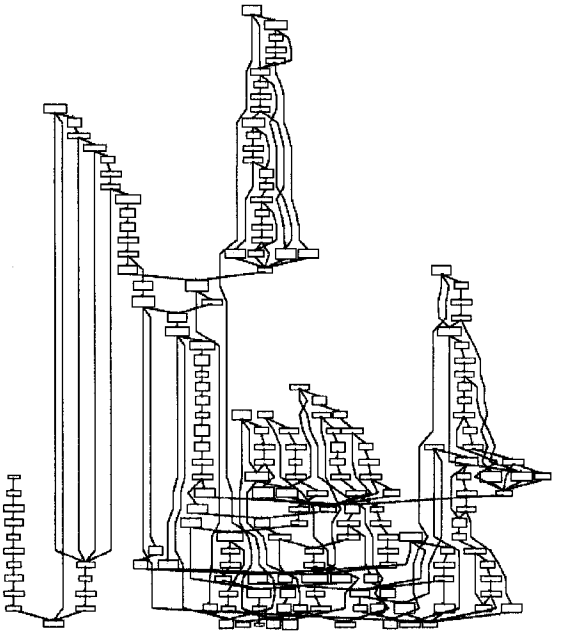

图 9 混淆后九宫格程序的 CFG

4 安全性分析及实验验证

安全性是混淆转换过程的重要属性之一。某混沌转换过程即使相当复杂，但其若极容易被攻击破解，则也无实际应用意义。安全性分析包括密码安全性、抗静态和抗动态攻击 3 方面。利用混沌理论构造的不透明谓词在以上 3 方面都具有较好的安全性。

为了验证混沌不透明谓词构造方法的密码安全性，实验中分别在测试程序中插入永真和永假混沌不透明谓词，通过统计改变密钥前后有效的混沌不透明谓词的个数来考察新方法的密码安全性。

如果插入永真的混沌不透明谓词，当改变之后由于生成的实数不能映射成 N 皇后问题的解而导致混沌不透明谓词的输出由 true 变为 false，那么插入的永真混沌不透明谓词将失效，会影响作用域范围内的程序的运行结果。当密钥改变之后生成的实数依然能映射 N 皇后问题的解，则混沌不透明谓词的输出仍然为 true，此时插入的混沌不透明谓词依然有效，不会影响其作用域范围内的程序的运行结果。反之，如果插入永假的混沌不透明谓词，则混沌不透明谓词的输出为 true 时插入的永假混沌不透明谓词失效，输出为 false 时插入的永假混沌不透明谓词有效。

实验中选择了 3 个测试程序，分别为九宫格程序、哈夫曼解压缩程序、DES 加密程序。初始参数和迭代初值为 $\mu=3.98$，$\beta=3.98$，$x_0=0.5$，$y_0=0.6$，改变之后的值为 $\mu=3.97$，$\beta=3.97$，$x_0=0.5$，$y_0=0.6$。分别在程序中插入 5 个永真和永

假混沌不透明谓词，它们将打印对应的混沌不透明谓词的输出，并以此方式统计改变密钥之后有效的不透明谓词的个数。结果如表4所列。

表4 改变密钥前后有效混沌不透明谓词的个数

测试程序	永真		永假	
	更改之前	更改之后	更改之前	更改之后
九宫格程序	5	0	5	5
哈夫曼解压缩程序	5	0	5	5
DES加密程序	5	0	5	5

从实验的结果可以看出，永真的混沌不透明谓词在密钥修改之后几乎全部失效，说明新的构造方法对密钥是敏感的，密码安全性较高。建议在混淆代码的过程中多使用永真的混沌不透明谓词。

结束语 本文研究了基于混沌理论的不透明谓词在代码混淆中的应用，对利用混沌不透明谓词技术混淆代码的整个过程进行了比较详细的描述。并通过实验从混淆转换前后程序的复杂度变化、程序控制流的复杂度变化两方面评价了混淆转换的有效性。对整个混淆转换系统的安全性进行了分析，并通过实验验证了整个混淆转换系统对密钥的高度敏感性，说明其密码安全性较强。如何通过程序来辅助用户做出最佳的混淆策略以达到运行效率和安全性的最佳平衡是下一步研究的方向。

参考文献

[1] 楼俊钢，江建慧，帅春燕，等. 软件可靠性模型研究进展[J]. 计算机科学，2010，37(9)：13-17

[2] 史扬，曹立明，王小平. 混淆算法研究综述[J]. 同济大学学报，2005，33(6)：813-819

[3] 付剑晶. 代码干扰变换在软件保护中的使用[J]. 计算机应用与软件，2008，25(4)：103-105

[4] 贾春福，王志，刘昕，等. 路径模糊：一种有效抵抗符号执行的二进制混淆技术[J]. 计算机研究与发展，2011，48(11)：2111-2119

[5] 袁征，冯雁，温巧燕，等. 构造一种新的混淆 Java 程序的不透明谓词[J]. 北京邮电大学学报，2007，30(6)：103-106

[6] Yuan Zheng, Wen Qiao-yan, Wu Wen-ling, et al. An ID-based watermarking scheme for java programs[C]// EUC Workshops, 2006. Berlin: Springer-Verlag, 2006: 848-857

[7] 袁春，钟玉琢，贺玉文. 基于混沌的视频流选择加密算法[J]. 计算机学报，2004，27(2)：257-262

[8] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms [M]. The MIT Press, 2009

[9] 赵玉洁，汤战勇，王妮，等. 代码混淆算法有效性评估[J]. 软件学报，2012，23(3)：700-711

[10] Collberg C. A Tool for the Study of Software Protection Algorithms[EB/OL]. http://sandmark.cs.arizona.edu, 2012-05-24

(上接第141页)

组件视图，可以方便地观察模型的组件以及组件关系。图8(b)为结果显示界面，显示输入信号和输出信号之间的关系，对于本实例，输入信号是指阶跃信号，输出是最右边运算放大器的输出信号，信号数据分别用蓝色线和红色线表示。

结束语 本文设计并实现了一个基于B/S结构的多领域可视化建模系统 WebMWorks，并提供了建模实例。相比单机环境下的建模系统，本系统具有以下特点：易于维护，方便升级部署；解决了可移动性和共享性差的问题；减少了客户端的压力；具有分布式、异地化的特性。

参考文献

[1] Modelica and the Modelica Association [EB/OL]. http://www.modelica.org, 2012-03-05

[2] 吴义忠，陈立平. 多领域物理系统的仿真优化方法[M]. 北京：科学出版社，2010：7-8

[3] 维基百科 Dymola[EB/OL]. http://en.wikipedia.org/wiki/Dymola, 2012-06-30

[4] 武汉天喻软件有限公司. 多领域物理系统建模与仿真平台 MWorks[EB/OL]. http://www.hustcad.com/NewContent.asp? ID=520, 2012-07-01

[5] 吴义忠，刘敏，陈立平. 多领域物理系统混合建模平台开发[J]. 计算机辅助设计与图形学学报，2006，18(1)：120-124

[6] 秦新燕. Modelica 语言在电路建模与仿真中的应用[J]. 湖北教育学报，2007，24(8)

[7] 莫雨峰，刘成良. 基于 Modelica 的航空发动机电气系统仿真分析研究[D]. 上海：上海交通大学，2004

[8] 董怡文，赵翼翔，陈新度. 基于 Modelica 的硫化机液压传动系统仿真[J]. 机床与液压，2010，38(17)：114-117

[9] 于红. 基于 Modelica 的球杆平衡系统建模仿真[J]. 机械工程与自动化，2011(1)：128-130

[10] 陈卫平，王波兴，黄运保. 基于多领域统一建模飞机起落架系统的仿真与分析[D]. 武汉：华中科技大学，2009

[11] 百度百科- Modelica [EB/OL]. http://baike.baidu.com/view/5623020.htm, 2012-01-10

[12] Torabzadeh-Tari M, Hossain Z M, Fritzson P. OMWeb-Virtual Web-based Remote Laboratory for Modelica in Engineering Courses [A] // Proceedings 8th International Modelica Conference, 2011[C]. Dresden, Germany: Linköping University Electronic Press, 2011: 153-159

[13] Shi Zheng-yin, Zhao Sheng-lin, Zhu Shan-an. An Internet-based Electrical Engineering Virtual Lab-Using Modelica for Unified Modeling [A] // 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN 2011), 2011 [C]. Xi'An, China: IEEE, 2011: 555-559

[14] Eissen S M, Stein B. Realization of Web-based simulation services[J]. Computers in Industry, 2006, 57(3): 261-271

[15] Duarte O. UN-VirtualLab: A Web simulation environment of OpenModelica models for educational purposes [A] // Proceedings 8th International Modelica Conference, 2011[C]. Dresden, Germany: Linköping University Electronic Press, 2011: 773-778

[16] 杨雷，邵诚. Web 环境下的 Simulink 仿真系统研究与开发[D]. 大连：大连理工大学，2010

[17] MacDonald M. Pro Silverlight 4 in C#[M]. New York: Apress, 2010: 260-278, 310-311