A watermarking scheme for structured programs

Lucila Bento\textsuperscript{1}
Davidson Boccardo\textsuperscript{2}
Raphael Machado\textsuperscript{3}
Vinícius Pereira de Sá\textsuperscript{1}
Jayme Szwarcfiter\textsuperscript{1,4}
\textsuperscript{1}Federal University of Rio de Janeiro,
\textsuperscript{2}Green Hat Information Security,
\textsuperscript{3}National Institute of Metrology, Quality and Technology
\textsuperscript{4}Estadual University of Rio de Janeiro

VII Latin American Workshop on Cliques in Graphs, November 2016
Overview

1. Motivation

2. Software Watermarking
   ▶ Definition
   ▶ Graph-based Software Watermarking

3. Structured Programs

4. Structured Watermarks
   ▶ Codec
   ▶ Properties

5. Conclusion
The insertion of watermarks into proprietary objects is a well-known means of discouraging piracy.
There exists a directed graph $P$ associated with every program, called control flow graph.
Graph-based Software Watermarking

Related works

2001 – Venkatesan, Vazirani and Sinha

Graph-based Software Watermarking
Related works

2001 – Venkatesan, Vazirani and Sinha


2003 – Collberg et al.

C. Collberg, S. Kobourov, E. Carter and C. Thomborson, Error-correcting graphs for software watermarking, 29th Workshop on Graph-Theoretic Concepts in Computer Science, WG’03, LNCS 2880 (2003), 156–167.
Graph-based Software Watermarking

Related works

2001 – Venkatesan, Vazirani and Sinha


2003 – Collberg et al.

C. Collberg, S. Kobourov, E. Carter and C. Thomborson, Error-correcting graphs for software watermarking, 29th Workshop on Graph-Theoretic Concepts in Computer Science, WG’03, LNCS 2880 (2003), 156–167.

2012 – Chroni and Nikolopoulos

The scheme comprises:

- \( encode(w) \rightarrow G \)
- \( decode(G) \rightarrow w \)
- \( embed(P, w, k) \rightarrow P' \)
- \( extract(P', k) \rightarrow w \)
Graph-based Software Watermarking

The scheme comprises

- \( encode(w) \rightarrow G \)
- \( decode(G) \rightarrow w \)
- \( embed(P, w, k) \rightarrow P' \)
- \( extract(P', k) \rightarrow w \)
Software Watermarking

Properties

Watermark Evaluation Properties

Stealthiness: A watermark should exhibit the same properties as the code around it so as to make detection difficult.

Small size: The size of the embedded watermark should be small.

Diversity: The ability to generate reasonably different watermarks to encode the same piece of information.

Resilience: A watermark should withstand a variety of attacks.
Watermark Evaluation Properties

- **Stealthiness:** A watermark should exhibit the same properties as the code around it so as to make detection difficult.
Software Watermarking

Properties

Watermark Evaluation Properties

▶ **Stealthiness**: A watermark should exhibit the same properties as the code around it so as to make detection difficult.

▶ **Small size**: The size $|P_w| - |P|$ of the embedded watermark should be small.
Watermark Evaluation Properties

- **Stealthiness**: A watermark should exhibit the same properties as the code around it so as to make detection difficult.

- **Small size**: The size $|P_w| - |P|$ of the embedded watermark should be small.

- **Diversity**: The ability to generate reasonably different watermarks to encode the same piece of information.
Software Watermarking

Properties

Watermark Evaluation Properties

- **Stealthiness**: A watermark should exhibit the same properties as the code around it so as to make detection difficult.

- **Small size**: The size $|P_w| - |P|$ of the embedded watermark should be small.

- **Diversity**: The ability to generate reasonably different watermarks to encode the same piece of information.

- **Resilience**: A watermark should withstand a variety of attacks.
There exist three main ways to attack a watermark in a software:

1. **Additive Attack**
   - $P_w \rightarrow P_w^{m} \rightarrow \text{Extract} \rightarrow ?$

2. **Subtractive Attack**
   - $P_w \rightarrow P' \rightarrow \text{Extract} \rightarrow ?$

3. **Distortive Attack**
   - $P_w \rightarrow P_w' \rightarrow \text{Extract} \rightarrow ?$
We are looking for the watermark that has the four main properties:

- Stealthiness
- Small size
- Diversity
- Resilience
Structured Programming

What is structured programming?

- Structured programming can be viewed as a method for the development and description of algorithms and programs.
- Basically, it consists of a top-down formulation of the algorithm, breaking it into blocks of modules.
- The technique constraints the description of the modules to contain only three basic control structures:
  - sequence
  - selection
  - iteration
- One of the early papers about structured programming was the seminal article by Dijkstra “Go-to statement considered harmful.”
Dijkstra Graphs

Definition

Statement Graphs

trivial

sequence

if

if-then-else

p-case

while

repeat
Definition

A Dijkstra graph (DG) is a graph with vertices labelled as $X$ or $R$, recursively defined as:

1. A trivial statement graph is a DG
2. Any graph obtained from a DG, by expanding some $X$-vertex of it into a non-trivial statement graph is also a DG. Furthermore, after expanding an $X$-labelled vertex $v$ into a statement graph, $v$ changes its label to $R$. 
Problem

- We are looking for a watermark that belong to the class of Dijkstra graphs to provide stealthiness.
Problem

- We are looking for a watermark that belong to the class of Dijkstra graphs to provide stealthiness.
Structured Watermarking

Encoding a Structured Watermark

Algorithm: encode(G)

Input: key \( \omega \)

Output: structured watermark \( G \)

Step 1 Let \( B \) be the binary representation of \( \omega \), and let \( n = |B| \).

Step 2 Create a watermark \( G(V, E) \) isomorphic to a Hamiltonian path.

Step 3 For each vertex \( v \), \( 1 \leq v \leq n \), add a edge from \( v \) to \( w \), where \( w \) is choose uniformly random between the \( V \) elements satisfying

- \( w \) is not an inner vertex of a cycle of \( G \), and
- \( v - w \) is odd if \( v \) is index of the 1 and even, otherwise.

Or if \( v \) is index of a bit 1 then creates a forward edge from \( v \) to \( v + 2 \), make \( v + 2 \) a “fill-in” vertex and add a vertex \( |V| + 1 \) in the Hamiltonian path. Else no back edge leaves \( v \).
Structured Watermarking

Encoding a Structured Watermark – Example

101110111

This is our binary, with $n$ bits.
Structured Watermarking
Encoding a Structured Watermark – Example

Step 1: Index its bits left to right starting from 1.
Structured Watermarking

Encoding a Structured Watermark – Example

Step 2: Build the directed path $P_{n+2}$ with vertex set $\{1, 2, \ldots, n + 2\}$. Now, for each index $i$ in the binary, we determine the destination of the back edge whose origin is $i$ (Step 3).
Structured Watermarking

Encoding a Structured Watermark – Example

101110111

1 2 3 4 5 6 7 8 9

No back edge here, of course. The first bit can only be a ‘1’ anyway.
Choose randomly among all possible back edges which can be drawn from the current vertex \( v \) and a vertex \( w \) such that

- \( w \) is not an inner vertex of a cycle that does not contain \( v \); and
- \( v - w \) is even (because the current bit is a ‘0’).

*The inner vertices of a cycle \( v_1, v_2, \ldots, v_d, v_1 \) are \( v_2, \ldots, v_{d-1} \).
Structured Watermarking

Encoding a Structured Watermark – Example

101110111
1  2  3  4  5  6  7  8  9

Choose randomly among all possible back edges which can be drawn from the current vertex \( v \) and a vertex \( w \) such that

- \( w \) is not an inner vertex of a cycle that does not contain \( v \); and
- \( v - w \) is odd (because the current bit is a ‘1’).

Or creates a forward edge from \( v \) to \( v + 2 \)
Choose randomly among all possible back edges which can be drawn from the current vertex $v$ and a vertex $w$ such that

- $w$ is not an inner vertex of a cycle that does not contain $v$; and
- $v - w$ is odd (because the current bit is a ‘1’).

Or creates a forward edge from $v$ to $v + 2$
Choose randomly among all possible back edges which can be drawn from the current vertex $v$ and a vertex $w$ such that

- $w$ is not an inner vertex of a cycle that does not contain $v$; and
- $v - w$ is odd (because the current bit is a ‘1’).
Choose randomly among all possible back edges which can be drawn from the current vertex $v$ and a vertex $w$ such that

- $w$ is not an inner vertex of a cycle that does not contain $v$; and
- $v - w$ is even (because the current bit is a ‘0’).
Choose randomly among all possible back edges which can be drawn from the current vertex $v$ and a vertex $w$ such that

- $w$ is not an inner vertex of a cycle that does not contain $v$; and
- $v - w$ is odd (because the current bit is a ‘1’).

Or creates a forward edge from $v$ to $v + 2$
Choose randomly among all possible back edges which can be drawn from the current vertex $v$ and a vertex $w$ such that

- $w$ is not an inner vertex of a cycle that does not contain $v$; and
- $v - w$ is odd (because the current bit is a ‘1’).
Choose randomly among all possible back edges which can be drawn from the current vertex $v$ and a vertex $w$ such that

- $w$ is not an inner vertex of a cycle that does not contain $v$; and
- $v - w$ is odd (because the current bit is a ‘1’).

Or creates a forward edge from $v$ to $v + 2$.
Structured Watermarking

Encoding a Structured Watermark – Example

101110111

1 2 3 4 5 6 7 8 9

Watermark is completed.
Algorithm: decode($G$)

**Input:** watermark $G$

**Output:** key $\omega$

**Step 1** Label the vertices of $G$ in ascending order as they appear in the Hamiltonian path of $G$.

**Step 2** Let $B$ be a bit array starting with a bit ‘1’ followed by $n - 1$ bits ‘0’.

**Passo 3** For each vertex $v$, for $2 \leq v \leq n$, if $v$ is not a “fill in” vertex and there is a vertex $w < v$ such that exists an edge from $v$ to $w$ and $v - w$ is odd or exists a forward edge leaves $v$ then $B[v] = 1$; otherwise $B[v] = 0$.

**Step 4** Return $\omega = \sum_{i=1}^{n} B[i] \cdot 2^{n-1}$. 
Structured Watermarking
Decoding a Structured Watermark – Example
First we find the (unique) Hamiltonian path in a backwards fashion, starting from the only vertex whose outdegree is zero. Label the vertices from 1 to $n$ along the path.
Structured Watermarking
Decoding a Structured Watermark – Example

The first bit is always a ‘1’.
The outdegree of the current vertex is 1 (no back edge leaves it). Therefore, it corresponds to a ‘0’ in the binary.
The back edge that leaves the current vertex reaches a vertex that is an odd distance apart from it along the Hamiltonian path. Thus, the current bit is a ‘1’.
Structured Watermarking
Decoding a Structured Watermark – Example

1011

A forward edge indicates a bit ‘1’.
The back edge that leaves the current vertex reaches a vertex that is an odd distance apart from it along the Hamiltonian path. Thus, the current bit is a ‘1’.
The back edge that leaves the current vertex reaches a vertex that is an even distance apart from it along the Hamiltonian path. Thus, the current bit is a ‘0’.
A forward edge indicates a bit ‘1’.
The back edge that leaves the current vertex reaches a vertex that is an odd distance apart from it along the Hamiltonian path. Thus, the current bit is a ‘1’.
The back edge that leaves the current vertex reaches a vertex that is an odd distance apart from it along the Hamiltonian path. Thus, the current bit is a ‘1’.
Structured Watermarking
Properties of Structured Watermarking

▶ **Efficiency**: both encoding and decoding algorithms can be implemented to run in linear time.
Structured Watermarking
Properties of Structured Watermarking

- **Efficiency**: both encoding and decoding algorithms can be implemented to run in linear time.

- **Small size**: The watermark has less than $n + \frac{n}{2}$ vertices and $2n + \frac{n}{2}$ edges, where $n$ is the bit-length of the key.
Structured Watermarking

Properties of Structured Watermarking

- **Efficiency:** both encoding and decoding algorithms can be implemented to run in linear time.

- **Small size:** The watermark has less than \( n + \frac{n}{2} \) vertices and \( 2n + \frac{n}{2} \) edges, where \( n \) is the bit-length of the key.

- **Diversity:** the encoding algorithm employs randomization to produce distinct watermarks for the same key upon different executions.
Resilience: there is a one-to-one correspondence between the edges of the watermark and the bits of the encoded key, hence distortive attacks are detected after the graph-to-key decoding process, and the correction of any flipped bits \( b \) up to some predefined number \( b \) is carried out by standard error-correction algorithms.
Structured Watermarking

Properties of Structured Watermarking

▶ Resilience: there is a one-to-one correspondence between the edges of the watermark and the bits of the encoded key, hence distortive attacks are detected after the graph-to-key decoding process, and the correction of any flipped bits \( b \) up to some predefined number \( b \) is carried out by standard error-correction algorithms.

▶ Stealthiness: The graphs produced by the encoding algorithm belong to the class of Dijkstra graphs. In short, when protecting structured software, the resulting watermarked CFG will belong to Dijkstra graphs.
Thank you!