# On the resilience of canonical reducible permutation graphs and their suitability for software watermarking[☆]

Lucila M. S. Bento[a,b], Davidson R. Boccardo[b], Raphael C. S. Machado[b], Vinícius G. Pereira de Sá[a], Jayme Luiz Szwarcfiter[a,b,c]

[a]*Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil*
[b]*Instituto Nacional de Metrologia, Qualidade e Tecnologia, Rio de Janeiro, Brasil*
[c]*COPPE Sistemas, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil*

**Abstract**

The ingenious graph-based software watermarking scheme proposed recently by Chroni and Nikolopoulos encodes integers as canonical reducible permutation graphs. Such graphs were claimed to withstand attacks in the form of a single edge removal. We introduce a linear-time algorithm which restores the original graph after ill-intentioned removals of $k \leq 2$ edges, therefore proving their claim. Furthermore, we prove that $k \leq 5$ general edge modifications (removals/insertions) can always be detected in polynomial time. Both bounds are tight. Our results reinforce the interest in regarding Chroni and Nikolopoulos's watermarks as a possible software security solution for numerous applications.

*Keywords:* distortive attacks, graph-based watermarking, linear-time algorithms, software security

[☆]Part of this paper was presented as an extended abstract entitled "Towards a provably resilient scheme for graph-based watermarking" at the 39th International Workshop on Graph Theoretic Concepts in Computer Science, WG 2013, and appeared in *Lecture Notes in Computer Science* **8165** (2013), 50–63.

*Email addresses:* `lucilabento@ppgi.ufrj.br` (Lucila M. S. Bento), `drboccardo@inmetro.gov.br` (Davidson R. Boccardo), `rcmachado@inmetro.gov.br` (Raphael C. S. Machado), `vigusmao@dcc.ufrj.br` (Vinícius G. Pereira de Sá), `jayme@nce.ufrj.br` (Jayme Luiz Szwarcfiter)

## 1. Introduction

Among the modern methods to fight the illegal reproduction of software, the embedding of digital watermarks is one which deserves attention. Roughly speaking, software watermarks are encodings of identification data hidden into a program. They allow for the timely retrieval of authorship and/or ownership information, and therefore discouraging piracy.

Soon after the creation of the first software watermark in 1996 by Davidson and Myrhvold [8], many interesting ideas have followed, including that of encoding a binary—the *identifier*—as a special digraph to be embedded into the software's control-flow graph, an idea which was patented by Venkatesan and Vazirani in 2006 [12]. *Graph-based* watermarking schemes have received a lot of attention ever since, and we give due emphasis to the contributions of Collberg *et al.* in a series of papers [5, 6, 7]. More recently, Chroni and Nikolopoulos presented an ingenious such scheme [3, 4], whereby the generated watermark graphs constitute a subclass of reducible flow graphs [9, 10, 11]. Such subclass possesses desirable features, among which its ease of implementation and its efficiency, since both encoding and decoding procedures run in linear-time. A third feature would be its alleged resilience to attacks. However, though its ability to withstand single edge removals has been briefly commented on in [4], not much has been proved in this sense.

We have previously [1, 2] given a formal characterization of the class of graphs produced by Chroni and Nikolopoulos's encoding function, which was referred to as the class of *canonical reducible permutation graphs*. We have also formulated a robust polynomial-time algorithm that retrieves the encoded identifier from a watermark with an arbitrary number $k \geq 0$ of deleted edges, whenever at all possible. In the present paper, we disclose the actual resilience of Chroni and Nikolopoulos's watermark by proposing a linear-time procedure which *always* succeeds in reconstituting a watermark from which $k \leq 2$ edges were removed, a bound which is best possible. Moreover, our results imply that $k \leq 5$ edge deletions and/or insertions can always be *detected* in polynomial time, a bound that is also tight.

This paper is organized as follows. In Section 2, we recall the watermark from Chroni and Nikolopoulos. In Section 3, we revisit some necessary definitions and previous results. In Section 4, we formulate linear-time algorithms to reconstruct the original graph and recover the encoded data even if two edges are missing. The proof of one of the central results in that section, namely Theorem 10, is somewhat involved, and we dedicate to it the whole Section 5. Section 6 concludes the paper with our final remarks.

Throughout the text, we let $V(G)$ and $E(G)$ respectively denote, as usual, the vertex set and edge set of a given graph $G$. Also, we let $N^+_G(v)$ and $N^-_G(v)$ be the sets of out-neighbors and in-neighbors of vertex $v$ in $G$, with $d^+_G(v)$ and $d^-_G(v)$ their respective sizes. If $J$ is a subset of either $V(G)$ or $E(G)$, then $G - J$ corresponds to the graph obtained from $G$ by the removal of $J$.

## 2. The watermark from Chroni and Nikolopoulos

We recall the encoding algorithm described in [4]. The index of the first element in all considered sequences is 1.

Let $\omega$ be a positive integer identifier, and $n$ the size of the binary representation $B$ of $\omega$. Let also $n_0$ and $n_1$ be the number of 0's and 1's, respectively, in $B$, and let $f_0$ be the index of the leftmost 0 in $B$. The extended binary $B^*$ is obtained by concatenating $n$ digits 1, followed by the one's complement of $B$ and by a single digit 0. We let $n^* = 2n + 1$ denote the size of $B^*$, and we define $Z_0 = (z^0_i)$, $i = 1, \ldots, n_1 + 1$, as the ascending sequence of indexes of 0's in $B^*$, and $Z_1 = (z^1_i)$, $i = 1, \ldots, n + n_0$, as the ascending sequence of indexes of 1's in $B^*$.

Let $S$ be a sequence of integers. We denote by $S^R$ the sequence formed by the elements of $S$ in backward order. If $S = (s_i)$, for $i = 1, \ldots, t$, and there is an integer $k \leq t$ such that the subsequence consisting of the elements of $S$ with indexes less than or equal to $k$ is ascending, and the subsequence consisting of the elements of $S$ with indexes greater than or equal to $k$ is descending, then we say $S$ is *bitonic*. If all $t$ elements of a sequence $S$ are distinct and belong to $\{1, \ldots, t\}$, then $S$ is a *permutation*. If $S$ is a permutation of size $t$, and, for all $1 \leq i \leq t$, the equality $i = s_{s_i}$ holds, then we say $S$ is *self-inverting*. In this case, the unordered pair $(i, s_i)$ is called a 2-*cycle* of $S$, if $i \neq s_i$, and a 1-*cycle* of $S$, if $i = s_i$. If $S_1, S_2$ are sequences (respectively, paths in a graph), we denote by $S_1||S_2$ the sequence (respectively, path) formed by the elements of $S_1$ followed by the elements of $S_2$.

Back to Chroni and Nikolopoulos's algorithm, we define $P_b = (b_i)$, with $i = 1, \ldots, n^*$, as the bitonic permutation $Z_0||Z^R_1$. Finally, the self-inverting permutation $P_s = (s_i)$ is obtained from $P_b$ as follows: for $i = 1, \ldots, n^*$, element $s_{b_i}$ is assigned value $b_{n^*-i+1}$, and element $s_{b_{n^*-i+1}}$ is assigned value $b_i$. In other words, the 2-cycles of $P_s$ correspond to the $n$ unordered pairs of distinct elements of $P_b$ that share the same minimum distance to one of the extremes of $P_b$, that is, the pairs $(p, q) = (b_i, b_{n^*-i+1})$, for $i = 1, \ldots, n$. Since the central index $i = n + 1$ of $P_b$ is the solution of equation $n^* - i + 1 = i$, element $b_{n+1}$ — and no other — will constitute a 1-cycle in $P_s$. We refer to such element of $P_s$ as its *fixed element*, and we let $f$ denote it.

The watermark generated by Chroni and Nikolopoulos's encoding algorithm [4] is a directed graph $G$ whose vertex set is $\{0, 1, \ldots, 2n + 2\}$, and whose edge set contains $4n + 3$ edges, to wit: a *path edge* $(u, u - 1)$ for $u = 1, \ldots, 2n + 2$, constituting a Hamiltonian path that will be unique in $G$, and a *tree edge* from $u$ to $q(u)$, for $u = 1, \ldots, n^*$, where $q(u)$ is defined as the vertex $v > u$ with the greatest index in $P_s$ to the left of $u$, if such $v$ exists, or $2n + 2$ otherwise. The rationale behind the name *tree edge* is the fact that such edges induce a spanning tree of $G \setminus \{0\}$.

Let us glance at an example. For $\omega = 349$, we have $B = 101011101$, $n = 9$, $n_0 = 3$, $n_1 = 6$, $f_0 = 2$, $B^* = 1111111110101000100$, $n^* = 19$, $Z_0 = (10, 12, 14, 15, 16, 18, 19)$, $Z_1 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 17)$, $P_b = (10, 12, 14, 15, 16, 18, 19, 17, 13, 11, 9, 8, 7, 6, 5, 4, 3, 2, 1)$, $P_s = (10, 12, 14, 15, 16, 18, 19, 17, 13, 1, 11, 2, 9, 3, 4, 5, 8, 6, 7)$ and $f = 11$. The watermark associated to $\omega$ presents, besides the path edges in the Hamiltonian path $20, 19, \ldots, 0$, the tree edges $(1, 13)$, $(2, 11)$, $(3, 9)$, $(4, 9)$, $(5, 9)$, $(6, 8)$, $(7, 8)$, $(8, 9)$, $(9, 11)$, $(10, 20)$, $(11, 13)$, $(12, 20)$, $(13, 17)$, $(14, 20)$, $(15, 20)$, $(16, 20)$, $(17, 19)$, $(18, 20)$ and $(19, 20)$.

## 3. Previous results

In [1], these authors proved a number of properties concerning the watermark from Chroni and Nikolopoulos and the special permutations they are associated to. We recall two of those properties, which are necessary for the recovering procedures described in Section 4. Since the proofs are omitted in the present text, the original numbers of the properties were provided for ease of reference.

Let $G$ be the watermark graph associated to an identifier $\omega$ of size $n$, and let $P_b$ and $P_s$ be, respectively, the bitonic and the self-inverting permutations dealt with during the construction of $G$ (see Section 2).

**Property 1 ([1], originally Property 3).** *The fixed element $f$ satisfies $f = n + f_0$, unless the identifier $\omega$ is equal to $2^k - 1$ for some integer $k$, whereupon $f = n^* = 2n + 1$.*

**Property 2 ([1], originally Property 8).** *For $u \leq 2n$, $(u, 2n+2)$ is a tree edge of watermark $G$ if, and only if, $u-n$ is the index of a digit* 1 *in the binary representation $B$ of the identifier $\omega$ represented by $G$.*

In that same paper [1], we introduced some definitions and results which will be necessary in this paper. We restate them in the remainder of this section for the sake of completeness. The proofs are again omitted, and we refer the reader to their original numbers in [1].

**Definition 3.** *A* self-labeling reducible flow graph *is a directed graph $G$ such that*

*(i) $G$ presents exactly one directed Hamiltonian path $H$, hence there is a unique labeling function $\sigma : V(G) \to \{0, 1, \ldots, |V(G)|-1\}$ of the vertices of $G$ such that the order of the labels along $H$ is precisely $|V(G)|, |V(G)|-1, \ldots, 0$; and,*

*(ii) considering the labeling $\sigma$ as in the previous item, $N^+_G(0) = \emptyset$, $N^-_G(0) = \{1\}$, $N^+_G(|V(G)|-1) = \{|V(G)|-2\}$, $|N^-_G(|V(G)|-1)| \geq 2$, and, for all $v \in V(G) \setminus \{0, |V(G)|-1\}$, $N^+_G(v) = \{v-1, w\}$, for some $w > v$.*

**Definition 4.** *The* representative tree $T$ *of a self-labeling reducible flow graph $G$ with Hamiltonian path $H$ has vertex set $V(T) = V(G) \setminus \{0\}$ and edge set $E(T) = E(G) \setminus E(H)$, where all edges are deprived of their orientation.*

A representative tree $T$ is always regarded as a *rooted* tree whose root is $|V(G)|-1$, Moreover, it is regarded as an *ordered* tree, that is, for each $v \in V(T)$, the children of $v$ are always considered according to an ascending order of their labels. For $v \in T$, we denote by $N^*_T(v)$ the set of descendants of $v$ in $T$.

Let $T$ be a representative tree. The *preorder traversal* $P$ of $T$ is a sequence of its vertices that is recursively defined as follows. If $T$ is empty, $P$ is also empty. Otherwise, $P$ starts at the root $r$ of $T$, followed by the preorder traversal of the subtree whose root is the smallest child of $r$, followed by the preorder traversal of the subtree whose root is the second smallest child of $r$, and so on. The last (rightmost) element of $P$ is also referred to as the rightmost element of $T$. The first element of the preorder traversal $P$ of a tree $T$ is always its root. If we remove the first element of $P$, the remaining sequence is said to be the *root-free* preorder traversal of $T$.

We can now recall the definition of canonical reducible permutation graphs. In [1], they were proved to correspond precisely to the graphs produced by Chroni and Nikolopoulos's encoding function.

**Definition 5.** *A canonical reducible permutation graph $G$ is a self-labeling reducible flow graph on $2n + 3$ vertices, for some integer $\geq 1$, such that the root-free preorder traversal of the representative tree of $G$ is a canonical self-inverting permutation.*

Let $T$ be the representative tree of some canonical reducible permutation graph $G$, and $P$ the root-free preorder traversal of $T$. We refer to the fixed element $f$ of $P$ also as the fixed element (or vertex) of both $G$ and $T$. Similarly, the 2-cyclic elements of $P$ correspond to *cyclic* elements (or vertices) of both $G$ and $T$. A vertex $v \in V(T) \setminus \{2n+2\}$ is considered *large* when $n < v \leq 2n+1$; otherwise, $v \leq n$ and $v$ is dubbed as *small*.

**Definition 6.** *A representative tree $T$ is a* Type-1 *tree if*

*(i)* $n+1, n+2, \ldots, 2n+1$ *are children of the root* $2n+2$ *in* $T$*; and*

*(ii)* $1, 2, \ldots, n$ *are children of* $2n$.

**Definition 7.** *A representative tree $T$ is a* Type-2 *tree relative to $f$ if*

*(i)* $n+1 = x_1 < x_2 < \ldots < x_\ell = 2n+1$ *are the children of* $2n+2$*, for some* $\ell \in [2, n-1]$*;*

*(ii)* $x_i > x_{i+1}$ *and* $x_i$ *is the parent of* $x_{i+1}$*, for all* $i \in [\ell, n-1]$*;*

*(iii)* $1, 2, \ldots, f-n-1$ *are children of* $x_n$*;*

*(iv)* $x_i = n+i$*, for* $1 \leq i \leq f-n-1$*;*

*(v)* $f$ *is a child of* $x_q$*, for some* $q \in [\ell, n]$ *satisfying* $x_{q+1} < f$ *whenever* $q < n$*; and*

*(vi)* $N^*_T(f) = \{f-n, f-n+1, \ldots, n\}$ *and* $y_i \in N^*_T(f)$ *has index* $x_{y_i} - f + 1$ *in the preorder traversal of* $N^*_T[f]$.

**Lemma 8 ([1], originally Lemma 21).** *If $y_r$ is the rightmost vertex of a Type-2 representative tree $T$ relative to some $f \neq 2n+1$, then $y_r$ is equal to the number $\ell$ of children of the root $2n+2$ in $T$.*

**Theorem 9 ([1], originally Theorem 22).** *A digraph $G$ is a canonical reducible permutation graph if, and only if, $G$ is a self-labeling reducible flow graph and*

*(i) the fixed element of $G$ is $2n+1$ and $G$ has a Type-1 representative tree; or*

*(ii) the fixed element of $G$ belongs to $[n+2, 2n]$ and $G$ has a Type-2 representative tree.*

## 4. Linear-time decoding ($k \leq 2$ missing edges)

In this section, we analyze the effects of a distortive attack against a watermark (i.e., a canonical reducible permutation graph) $G$ from which $k \leq 2$ edges were removed. Note that the unique Hamiltonian path $H$ of $G$ may have been destroyed by the attack. The knowledge of $H$ is crucial for determining the labels of the vertices (they range from $2n+2$ to 0 along $H$). Our first task is therefore to determine whether any path edges are missing from $G$, so we can restore $H$ and label the vertices accordingly.

### *4.1. Reconstructing the Hamiltonian path*

The algorithm given in pseudocode as Algorithm 1 retrieves the unique Hamiltonian path $H$ of a (possibly damaged) watermark $G'$, that is, a graph isomorphic to a canonical reducible permutation graph $G$ minus $k \leq 2$ edges. It employs two subroutines presented separately: *plug_next_subpath* and *validate_labels*. The algorithm itself is straightforward. It basically builds Hamiltonian path candidates for $G$ (possibly by reinserting some edges) and tests whether the vertex labeling implied by each such candidate satisfies some conditions. It returns the one and only candidate which passes the test.

The procedure *plug_next_subpath*, given as Algorithm 2, takes as input a graph $G'$, a path $Q$ with $V(Q) \cap V(G) = \emptyset$, and an output list $\mathcal{H}$, where (restored) Hamiltonian path candidates of $G'$ will be placed after being concatenated to (the left of) a copy of $Q$. It starts by determining a set $S$ of *subpath heads*. This set comprises every vertex $s \in V(G')$ satisfying $d^+_{G'}(s) = 0$, in case $Q = \emptyset$, or $d^+_{G'}(s) \leq 1$, otherwise. Then it computes the collection of

**Algorithm 1** *reconstruct_Hamiltonian_path*$(G')$

```
input: a damaged watermark $G'$ with $2n+3$ vertices and
       two missing edges
output: the unique Hamiltonian path $H$ in the original watermark $G$

1. Let $V_0$ be the set of all vertices with degree zero in $G'$.

2. Let $\mathcal{H}$ be the set of all Hamiltonian path candidates obtained by
   a call to plug_next_subpath$(G', V_0, \emptyset)$.

3. for each Hamiltonian path candidate $H \in \mathcal{H}$ do
       if validate_labels$(G', H)$ then
           return $H$
```

all *maximal backward-unbifurcated paths* of $G'$ reaching $S$ (or $S$-bups). An $S$-*bup* of $G'$ is a path $v_j, v_{j-1}, \ldots, v_1$ such that

- $v_1 = s$, for some $s \in S$;
- $(v_k, v_{k-1}) \in E(G')$, for $2 \leq k \leq j$; and
- the in-degree of $v_k$ in $G' - \{v_1, \ldots, v_{k-1}\}$ satisfies

$$d^-_{G'-\{v_1,\ldots,v_{k-1}\}}(v_k) = \begin{cases} 1, & \text{if } 1 \leq k \leq j-1; \\ 0, & \text{if } k = j. \end{cases}$$

In other words, starting from some subpath head $s = v_1$, the procedure builds a directed path $Q$ in backwards fashion by concatenating an in-neighbor of $v_k$ to the left of $v_k$, for $k \geq 1$, whenever the in-degree of $v_k$ is 1 in the graph induced by all vertices which have not yet been incorporated to the path. It carries on iteratively this way until, for some $j$, the in-degree of $v_j$ in the aforementioned graph is either zero, whereupon it adds the path so obtained to a list of $S$-bups, or greater than one, whereupon it discards the current path. The rationale behind it is that a backward bifurcation on $v_j$ means there are two vertices, say $u$ and $w$, which have not yet been added to the path, both of which are in-neighbors of $v_j$. Since at most one of them, say $u$, may be the tail of a path edge pointing to $v_j$ in $Q$, the other one, $w$, will be

**Algorithm 2** *plug_next_subpath*$(G', Q, \mathcal{H})$

```
input: a graph $G'$, a path $Q$ with $V(Q) \cap V(G') = \emptyset$,
       and an output list $\mathcal{H}$
output: an updated $\mathcal{H}$ containing all $Q'||Q$
        where $Q$ is a Hamiltonian path of $G'$ (plus $k \geq 0$ extra edges)
        ending at a vertex of degree $d \leq 1$ (or $d = 0$, if $Q$ is empty)

1. Let $S \leftarrow \{s \in V(G') \colon d^+_{G'}(s) = 0\}$.
   if $Q \neq \emptyset$ then $S \leftarrow S \cup \{s \in V(G) \colon d^+_{G'}(s) = 1\}$

2. for each $s \in S$ do
       $v \leftarrow s$
       $Q' \leftarrow s$
       while $|N^-_{G'-(V(Q')-\{v\})}(v)| = 1$ do
           $v \leftarrow$ the unique element in $N^-_{G'-(V(Q')-\{v\})}(v)$
           $Q' \leftarrow v||Q'$
       if $|Q'| = |V(G')|$ then
           $\mathcal{H} \leftarrow \mathcal{H} \cup \{Q'||Q\}$
       else if $|N^-_{G'-(V(Q')-\{v\})}(v)| = 0$ then
           plug_next_subpath$(G' - V(Q'), Q'||Q, \mathcal{H})$
       else discard $Q'$  //  a backward bifurcation was found
```

the tail of a tree edge pointing to $v_j$, which is not acceptable since $w$ will be to the left of $v_j$ in the path. Whichever the case, the algorithm starts anew with another subpath head $s \in S$ until all of them have been considered and the list of $S$-bups is fully populated. Finally, it appends each $S$-bup $Q'$, one at each time, to the left of $Q$ (by adding a *plausible path edge* $e \notin E(G')$ from the rightmost vertex in $Q'$ to the leftmost vertex in $Q$) and performs one of two possible actions:

- if $V(Q') = V(G')$ (i.e., if $Q'$ is a Hamiltonian path of $G'$), than it adds the new path $Q'||Q$ to the output list $\mathcal{H}$;
- otherwise, it makes a recursive call to *plug_next_subpath* with parameters $G' - V(Q'), Q'||Q$, and the output list $\mathcal{H}$.

When all $S$-bups have been considered, it returns $\mathcal{H}$.

If $H$ is a path, then we indicate the $j$th element of $H$ (from right to left, starting at $j = 0$) by $H[j]$.

The second subroutine invoked by Algorithm 1 is called *validate_labels*, shown in pseudocode as Algorithm 3. It takes as parameters a watermark $G'$ (with two missing edges) and a candidate Hamiltonian path $H$. First, it determines the set $H^* = E(H) \setminus E(G')$ of the $k \leq 2$ plausible path edges that were required by $H$. It then checks whether it is possible to obtain a valid canonical reducible permutation graph $G$ through the insertion of $H^*$ and some set of $2 - k$ tree edges into $G'$. It does so by testing the following necessary conditions, where $T$ denotes the representative tree of $G$:

(1) vertices $H[2n+1]$ and $H[n+1]$ must be in-neighbors of $H[2n+2]$ in $G$;

(2) the out-degree of vertices $H[2n+1], \dots, H[1]$ must be 2 in $G$, and the out-degree of $H[2n+2]$ must be 1;

(3) the number of tree edges that would have to be inserted into $G$ so that the two previous conditions are met must not exceed $2 - |H^*|$; and, finally,

(4) if vertices $H[1]$ and $H[n]$ are not siblings in $T$, then vertex $H[1]$ must be a child of the $n$th descendant of the root $H[2n+2]$ which is a large vertex (i.e., the $n$th descendant of the root, counted right to left, among those whose indexes in $H$ are greater than or equal to $n+1$); moreover, the index in $H$ of the rightmost vertex in the preorder traversal of $T$ must correspond to the number of children of $2n + 2$ in $T$.

The first condition above appears in the definition of both Type-1 and Type-2 representative trees (Definitions 6 and 7), hence its necessity comes directly from Theorem 9. The second condition is due to Definition 5 and from the second property in the definition of self-labeling reducible flow graphs (Definition 3). The third condition obviously comes from the fact that 2 edges were removed from $G$, and $|H^*|$ edges have already been (re-)inserted at this point. Finally, the last condition is due to property (iii) in Definition 7 and to Lemma 8. It certainly applies to Type-2 representative trees only, which is precisely the case where vertices $H[1]$ and $H[n]$ are not siblings in the representative tree, by definition. We remark that, if $H$ is indeed the unique Hamiltonian path of a canonical reducible permutation graph $G$, then, for all $v \in V(G)$, the canonical label $v$ satisfies $v = H[v]$.

**Algorithm 3** *validate_labels*$(G', H)$

```
input: a graph $G'$, with $|V(G')| = 2n+3$,
       and a Hamiltonian path candidate $H$, with $|E(H) \setminus E(G')| \leq 2$
output: True, if the labeling of $V(G')$ implied by $H$ is valid; False, otherwise

1. Label the vertices of $G'$ in such a way that $H = 2n+2, 2n+1, \ldots, 0$.
   Let $H^* \leftarrow E(H) \setminus E(G')$, and insert $H^*$ into $G'$ obtaining $G''$.
   Let also $F$ be the forest obtained from $G''$ by the removal of all (path)
   edges in $H$, as well as the isolated vertex 0, and let missing_edges $\leftarrow 0$.

2. for each $v \in \{n+1, 2n+1\}$ do
       if $(v, 2n+2) \notin E(G'')$ then
           if $d^+_{G''}(v) = 2$ then return False
           $E(G'') \leftarrow E(G'') \cup \{(v, 2n+2)\}$; missing_edges += 1
   for each $v \in \{1, \ldots, 2n+1\}$ do
       if $d^+_{G''}(v) < 2$ then missing_edges += 1
   if missing_edges $> 2 - |H^*|$ then return False

3. if vertices 1 and $n$ are siblings in $F$ then
       Let $r$ be the rightmost vertex in the preorder traversal of $F$.
       if $r > d^-_{G''}(2n+2)$ + missing_edges then return False
       Let $x$ be the length of the unique path from 1 to $2n+2$ in $F$.
       if $r + x - 2 \neq n$ then return False

4. return True
```

**Theorem 10.** *Algorithm 1 correctly retrieves the original, unique Hamiltonian path from a canonical reducible permutation graph on $2n+3$ vertices from which $k \leq 2$ edges were removed. It runs in $O(n)$ time.*

The proof of Theorem 10 is somewhat involved and unfortunately demands some case analysis. We therefore postpone it until Section 5.

### *4.2. Determining the fixed vertex*

Suppose the watermark $G$ has been attacked, which resulted in a damaged watermark $G'$, where two unknown edges are missing. Now we shall recognize the fixed vertex of the original watermark, given the damaged one. Getting to know the fixed vertex of $G$ will play a crucial role in retrieving the missing tree edges and consequently restoring the original identifier $w$ encoded by $G$.

We describe some characterizations that lead to an efficient computation of the fixed vertex $f$ of $G$. Let $T$ be the representative tree of the original watermark $G$. We consider the case where the two edges that have been removed belong to $T$. Denote by $F$ the forest obtained from $T$ by the removal of two edges. First, we consider the case $f = 2n + 1$.

**Theorem 11.** *Let $F$ be a forest obtained from the representative tree $T$ by removing two edges, where $n > 2$. Then $f = 2n + 1$ if, and only if,*

1. *vertex $2n + 1$ is a leaf of $F$; and*
2. *the $n$ small vertices of $G'$ are children of $2n$ in $F$, with the possible exception of at most two of them, in which case they must be isolated vertices.*

*Proof:* From Theorem 9, we know that, when $f = 2n+1$, $f$ is the rightmost vertex of $T$, hence a leaf of $F$, implying the necessity of condition (i). Again by Theorem 9, the small vertices of $T$ must immediately follow the rightmost cyclic vertex of $T$, namely $2n$. Since two edges have been deleted from $T$, it follows that all small vertices are children of $2n$ in $F$, with the possible exception of at most two of them, which then became isolated vertices, so condition (ii) is also necessary.

Conversely, suppose conditions (i) and (ii) hold, and assume $f \neq 2n + 1$. Then the second case covered by Theorem 9 applies for $T$. If $f = 2n$, we know from $n > 2$ that vertex $2n + 1$ has at least 3 children in $T$, making it impossible for $2n + 1$ to become a leaf of $F$ by the removal of only two edges, therefore contradicting condition (i). If, on the other hand, $f < 2n$, then, again by Theorem 9, vertex $2n + 1$ cannot have any small children, contradicting condition (ii). Therefore $f > 2n$, implying $f = 2n + 1$. $\square$

Next, we characterize the case $f < 2n+1$. Figure 1 helps to visualize the three conditions of the theorem.

**Theorem 12.** *Let $F$ be a forest obtained from the representative tree $T$ of watermark $G$ by removing two of its edges, and let $x \leq 2n$ be a large vertex of $T$ which is not a child of $2n + 2$. Then $x$ is the fixed vertex $f$ of $G$ if, and only if,*

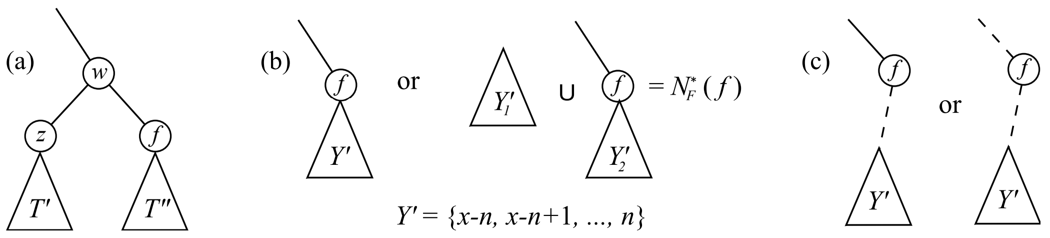

Figure 1: (a–c) Conditions (i), (ii) and (iii) of Theorem 12, respectively.

*(i) the large vertex $x$ has a sibling $z$ in $F$, and $x > z$; or*

*(ii) the subset of small vertices $Y' \subset Y$, $Y' = \{x-n, x-n+1, \ldots, n\}$ can be partitioned into at most two subsets $Y'_1, Y'_2$, such that $\emptyset \neq Y'_1 = N^+_F(x)$ and $Y'_2$ is the vertex set of one of the trees which form $F$; or, whenever the previous conditions do not hold,*

*(iii) the large vertex $x$ is the rightmost vertex of one of the trees of $F$, while the rightmost vertices of the remaining trees are all small vertices.*

*Proof:* For the sufficiency of condition (i), let $x$ be a large vertex of $F$, $z$ a sibling of $x$ in $F$ and $x_q$ their parent. By Theorem 9, the only large vertex of $T$ which is not a child of $2n+2$ and has some sibling $z$ is precisely the fixed vertex $f$. Clearly, the removal of edges of $T$ cannot create new vertices having this property. Furthermore, $x_q \notin \{x_n, 2n+1\}$ implies that $f$ has a unique sibling $x_{q+1}$, hence $f > x_{q+1}$ according to the ascending order of siblings in $F$, whereas $x_q \in \{x_n, 2n+1\}$ implies every sibling $y$ of $f$ is a small vertex, hence $f > y$. Consequently, $x = f$.

Now suppose condition (ii) holds. First, assume that $Y'_2 = \emptyset$. In this case, $Y' = Y'_1 = \{x-n, x-n+1, \ldots, n\} = N^*_F(x)$. Again, according to Theorem 9, we can locate a unique vertex $f$ fulfilling this property, implying $x = f$. In addition, when $Y'_2 \neq \emptyset$, we can again select a unique vertex $f$, where $N^*_F(f) \cup Y'_2 = Y'$. Thus, $x = f$ indeed.

Finally, assume neither condition (i) nor condition (ii) hold. Because (i) is not satisfied, we have that either $x_q \notin \{x_n, 2n+1\}$, and the edge from $x_q$ to one of its children has been deleted; or $x_q \in \{x_n, 2n+1\}$, and the edge $(x_q, f)$ has been deleted. Additionally, since (ii) is not satisfied, $f$ must have a unique child $y$, and the edge $(f, y)$ has also been removed. Next, assume that, in such a context, condition (iii) is verified. For the sake of contradiction, suppose the theorem is false, so that $x \neq f$. Since $x \neq 2n+1$

and $x$ is not a child of $2n+2$, it follows that it must be a descending vertex, whereupon the fact that $x$ is the rightmost vertex of the tree of $F$ containing it implies that $x$ is a leaf of $F$. Now the latter implies that the edge $(x_q, x)$ of $T$ has been removed, where $x_q$ is the parent of $x$ in $T$. Because condition (i) is not satisfied, at least one edge has been removed from $T$, and because condition (ii) is not satisfied, at least one more edge has been deleted from $T$. Since no more than two edges overall have been removed, we conclude that the assumption is false, and therefore, here again, $x = f$.

Conversely, assume that $x$ is the large vertex of $F$ satisfying $x = f$. We prove that condition (i) or condition (ii) holds, otherwise condition (iii) is satisfied.

Let $x_q$ be the parent of $x = f$ in $T$. If $x_q$ has at least two children in $F$, then $f$ is larger than its siblings, by Theorem 9, and condition (i) holds. Alternatively, if $f$ is not a leaf of $F$, then the set $Y' = \{x-n, x-n+1, \ldots, n\}$ either satisfies $Y' = N^*_F(f)$ or it can be split into two subsets $Y'_1 \cup Y'_2 = Y'$, where $Y'_1 = N^*_F(f)$ and $Y'_2$ is the vertex set of one of the trees of $F$. In this situation, condition (ii) holds. Assume, next, that neither condition (i) nor condition (ii) hold. Then the parent $x_q$ of $f$ in $T$ has at most one child in $F$, whereas $f$ has no children. The latter implies that $f$ is the rightmost vertex of the tree of $F$ containing it. Since $x_q \neq f$, we know that no more than two edges have been deleted from $T$, hence no large vertex other than $f$ can be a leaf of $F$. Consequently, condition (iii) holds, completing the proof. $\square$

The above theorems lead to an algorithm that efficiently finds the fixed vertex of watermark $G$ (see Algorithm 4). The input is the forest $F$, obtained from the representative tree $T$ of $G$ by the removal of two edges. First, the algorithm checks whether $f = 2n+1$. By Theorem 11, it suffices to verify whether $2n+1$ is a leaf of $F$ and all small vertices are children of $2n$, except possibly two, which must be isolated vertices. If this is not the case, then the algorithm proceeds to determining $f$ knowing that $f < 2n+1$. Basically, such task consists in checking conditions (i), (ii) and (iii) of Theorem 12, which can be done in a straightforward manner.

Steps 1 and 3 can be computed easily in linear time. As for Step 2, observe that there are at most two large vertices $x$ of $F$ that may satisfy the condition of having only small children. Consequently, the tests in Step 2 apply to at most two candidates $x$, hence the entire algorithm runs in $O(n)$ time.

**Algorithm 4** *find_f*$(F)$

```
input: a forest $F$ (a representative tree with two missing edges)
output: the fixed element $f \leq 2n+1$

1. if $F$ contains a large vertex $x$ having a sibling $z$, then
       return $f := max\{x, z\}$

2. for each large vertex $x$ of $F$ satisfying $N_F(x) \neq \emptyset$
       for each small $y \in N_F(x)$
           $Y' \leftarrow \{x-n, x-n+1, \ldots, n\}$
           if $(N^*_F(x) = Y'$ or $N^*_F(x) \subset Y')$ and
              $(Y' \setminus N^+_F(x)$ is the vertex set of a tree of $F)$ then
                return $f := x$

3. Find the preorder traversals of the 3 trees of $F$, and let $f$ be the
   unique vertex that is both large and the rightmost element of the
   preorder traversal of some tree of $F$.
   return $f$
```

## *4.3. Determining the root's children*

After having identified the fixed vertex of the watermark, we are almost in a position to determine the tree edges that have been removed.

Observe that, when $f = 2n+1$, the task is trivial, since, in this case, by Theorem 9, there can be only one canonical reducible permutation graph $G$ relative to $n$. Such graph is precisely the one with a Type-1 representative tree $T$, which is unique for each $n > 2$ (cf. Property 1 of canonical reducible permutation graphs, in Section 2). By definition, the root-free preorder traversal of a Type-1 representative tree, when $f = 2n+1$, is $n+1, n+2, \ldots, 2n, 1, 2, \ldots, n, 2n+1$.

We therefore want to determine the children of $2n+2$ restricted to the case where $f < 2n+1$. Let $G$ be a watermark, $T$ its representative tree and $F$ the forest obtained from $T$ by the removal of two edges. As usual, $f$ stands for the fixed vertex of $T$, $X$ is the set of large vertices other than $2n+2$, and $X_c = X \setminus \{f\}$. Finally, denote by $A \subseteq X_c$ the subsets of ascending large cyclic vertices of $T$, which we shall refer to simply as the *ascending* vertices, and denote by $D$ the set $D = X_c \setminus A$ of descending large cyclic vertices of

**Algorithm 5** *find_ascending_large_vertices*$(F)$

```
input: a forest $F$ (a representative tree with two missing edges),          and
the set $X_c$ of the large cyclic vertices in $F$
output: the children $A$ of the root $2n+2$ of the representative tree

1. if $F[X_c] \cup 2n+2$ is connected then
       return $A := N_F(2n+2)$

2. if $F[X_c] \cup 2n+2$ contains no isolated vertices then
       return $A := N_F(2n+2) \cup 2n+1$

3. if $F[X_c] \cup 2n+2$ contains two isolated vertices $x, x'$ then
       return $A := N_F(2n+2) \cup \{x, x'\}$

4. if $F[X_c] \cup 2n+2$ contains a unique isolated vertex $x$ then
       if $|N_F^*(f)| = 2n - f + 1$ then
           let $y_r$ be the rightmost vertex of $N_F^*(f)$
           if $|N_F(2n+2)| < y_r$ then
               return $A := N_F(2n+2) \cup \{x, 2n+1\}$
           else
               return $A := N_F(2n+2)$
       else
           return $A := N_F(2n+2) \cup \{x\}$
```

$T$, or simply the *descending* vertices. Given the forest $F$ and its fixed vertex $f$, Algorithm 5 computes the set $A$, which, as we recall from the proof of Theorem 9, corresponds precisely to the children of the root $2n+2$.

It is easy to conclude that the above algorithm can be implemented in $O(n)$ time. Now we prove its correctness.

**Theorem 13.** *Algorithm 5 correctly computes the set $A$ of ascending vertices of $T$.*

*Proof:* We follow the different conditions checked by the algorithm. Assume $F[X_c] \cup \{2n+2\}$ is connected. Then $N_T(2n+2) = N_F(2n+2)$, implying $A = N_F(2n+2)$, and the algorithm is correct if it terminates at Step 1.

Assume $F[X_c] \cup \{2n+2\}$ is disconnected, but has no isolated vertices. Then either $N_F(2n+2) = N_F(2n+2)$ or the edge $(2n+2, 2n+1)$ was one of those that might have been removed from $T$. In any of these situations, we can write $A = N_F(2n+2) \cup 2n+1$, implying that the algorithm is also correct if it terminates at Step 2.

Assume $F[X_c] \cup \{2n+2\}$ contains two distinct isolated vertices $x, x'$. The only possibility is $x, x' \in N_T(2n+2)$. So, the action of constructing $A$ as the union of $x, x'$ and $N_F(2n+2)$ assures correctness, whenever the algorithm terminates at Step 3.

The last situation is $F[X_c] \cup \{2n+2\}$ containing a unique isolated vertex $x$. We consider the following alternatives. If $|N_F^+(f)| = 2n - f + 1$, it implies that $N_T^*(f) = N_F^*(f)$, because the set of descendants of $f$ in $T$ comprises exactly $y_{f_0}, y_{f_0+1}, \ldots, y_n$, The number of such descendants of $f$ is therefore $n - f_0 + 1$, which, by Property 1 of canonical reducible permutation graphs, is equal to $2n - f + 1$. Now, by Theorem 9, $|N_T(2n+2)| = y_r$, where $y_r$ is the rightmost vertex of $N_F^*(f)$. In this situation, $|N_F(2n+2)| < y_r$ implies that $x$ necessarily belongs to $N_F(2n+2)$. In addition, edge $(2n+2, 2n+1)$ might also have been deleted from $T$, since a single edge deletion suffices to turn $x$ into an isolated vertex. Observe, on the other hand, that isolating a large vertex which is not a child of $2n+2$ requires the removal of at least two edges, provided $n > 2$. Thus, $A = N_F(2n+2) \cup \{x, 2n+1\}$, and the algorithm is correct. In case $|N_F(2n+2)| = y_r$, we know that $N_T(2n+2) = N_F(2n+2)$, hence $A = N_F(2n+2)$, ensuring correctness. Finally, when $|N_F^*(f)| \neq 2n - f + 1$, it means some edge inside the subtree rooted at $f$ has been deleted from $T$. In this case, the isolated vertex $x$ is necessarily a child of $2n+2$ in $T$, implying $A = N_F(2n+2) \cup \{x\}$, and the algorithm is correct. □

## *4.4. Retrieving the missing edges*

Once we know the set of ascending vertices, it is simple to restore the entire tree $T$. Basically, given sets $A$ and $X_c$, we obtain the set of $D$ of descending vertices. Then, by sorting $A$ and $D$ accordingly, we can locate all the large cyclic vertices in $T$, using the model given by Theorem 9. We then place $f$ in $T$, such that its parent $x_q$ is smallest cyclic vertex that is larger than $f$. Finally, we place the small vertices. Vertices $\{1, 2, \ldots, f-n-1\}$ are all children of $x_n$. The remaining small vertices $\{f-n, f-n+1, \ldots, n\}$ are descendants of $f$ and their exact position in $T$ can be obtained as follows. For each $y \in \{f-n, f-n+1, \ldots, n\}$, we find its position in the preorder traversal $P$ of $T$ by determining the large vertex $x$ whose position in the bitonic sequence of the cyclic large vertices is exactly $y$. Then $y$ must be the

**Algorithm 6** *retrieve_preorder_traversal*$(T, f, A, X_c)$

```
input: the fixed vertex $f$, the set $A$ of ascending vertices,
       and the set $X_c$ of large cyclic vertices in a representative tree $T$
output: the preorder traversal $P$ of $T$

1. Let $D \leftarrow X_c \setminus A$.
2. The initial vertices of $P$ are those of $A$ in ascending order,
   followed by those of $D$, in descending order.
   Now, subsequently place in $P$ the small vertices $1, \ldots, f - n - 1$,
   in this exact order, immediately after the last descending
   vertex $x_n \in D$. Then place $f$ as to immediately follow $f - n - 1$.
3. for each small vertex $y \in \{f - n, f - n + 1, \ldots, n\}$
       Let $P[y]$ be the (large) vertex $x$ whose index in $P$ is $y$.
       Place $y$ at position $x$ in $P$, i.e., satisfying $P[x] = y$.
4. return $P$
```

$x$th vertex in the root-free preorder traversal of $T$. Finally, the position of $f$ in $P$ is clearly equal to $f$.

The details are given in Algorithm 6, which computes the preorder traversal $P$ of $T - \{2n + 2\}$ in time $O(n)$. Such procedure assures the complete retrieval of $T$ and therefore we are able to restore the watermark $G$ in full.

### *4.5. Linear-time decoding with $k \leq 2$ missing edges*

We can now formulate a new, resilient decoding algorithm, presented as Algorithm 7. If the input watermark presents $k \leq 2$ missing edges, the algorithm fixes it prior to running the actual decoding step, which is a simple call to the decoding algorithm presented by these authors in [1].

**Theorem 14.** *Algorithm 7 retrieves the correct identifier, encoded in a watermark with up to two missing edges, in linear time.*

*Proof:* Since the final step of the algorithm runs in linear time, its overall time complexity relies on the fact that Algorithms 1–6 run in linear time themselves, as proved earlier in the text. The correctness of the algorithm

**Algorithm 7** *resilient_decode*$(G)$

```
input: a watermark $G$ with $2n+3$ vertices and $0 \leq k \leq 2$ missing edges
output: the identifier $\omega$ encoded by $G$

1. Let $k \leftarrow |E(G)| - (4n+3)$.
2. **if** $k > 2$ **then** report the occurrence of $k$ edge removals and halt.
3. **if** $0 < k \leq 2$ **then** proceed to the reconstitution of the watermark
   (see Section 4, Algorithms 1–6).
4. **return** $\omega$ via the linear-time decoding procedure from [1].
```

follows from the ability to reconstruct the original watermark when $k \leq 2$ edges have been removed. □

**Corollary 15.** *Distortive attacks in the form of $k$ edge modifications (insertions/deletions) against canonical reducible permutation graphs $G$, with $|V(G)| = 2n+3$, $n > 2$, can always be detected in polynomial time, if $k \leq 5$, and also recovered from, if $k \leq 2$. Such bounds are tight.*

*Proof:* From Theorem 14, we know that, for $n > 2$, there are no two watermarks $G_1, G_2$, with $|V(G_1)| = |V(G_2)| = 2n+3$, such that

$$|E(G_1) \setminus E(G_2)| = |E(G_2) \setminus E(G_1)| \leq 2,$$

otherwise it would not always be possible to recover from the removal of up to two edges. Thus, for $n > 2$, any two canonical reducible permutation graphs $G_1, G_2$ satisfy

$$|E(G_1) \setminus E(G_2)| = |E(G_2) \setminus E(G_1)| \geq 3, \tag{1}$$

hence $G_1$ cannot be transformed into $G_2$ by less than 6 edge modifications. Since the class of canonical permutation graphs can be recognized in polynomial-time in light of the characterization given in Theorem 9, and since any number $k \leq 5$ of edge modifications made to a graph $G$ of the class produces a graph $G'$ that does not belong to the class, all distortive attacks of such magnitude ($k \leq 5$) can be detected. Now, for $k = 2$, we have three possibilities:

(i) two edges were removed;

(ii) two edges were inserted;

(iii) one edge was removed and one edge was inserted.

If case (i) applies, Theorem 14 guarantees that the original graph can be successfully restored. If case (ii) or case (iii) apply, then a simple algorithm in which all possible sets of two edge modifications are attempted against the damaged graph $G'$ suffices to prove that the original graph $G$ can be restored in polynomial time, since, as we already know, exactly one such set shall turn $G'$ into a canonical reducible permutation graph. The case $k = 1$ is simpler and can be tackled in analogous manner.

It remains to show that such bounds are tight. For $n > 3$, let $\omega_1 = 2^{n-1}$ and $\omega_2 = 2^{n-1} + 1$. It is easy to see that the graphs $G_1, G_2$ encoding $\omega_1, \omega_2$ satisfy (1) with equality. Indeed, by construction, the self-inverting permutation associated to $\omega_1$ is $(n+1, 2n+1, 2n, 2n-1, 2n-2, \ldots, n+3, 1, n+2, n, n-1, \ldots, 4, 3, 2)$, and the self-inverting permutation associated to $\omega_2$ is $(n+1, 2n, 2n+1, 2n-1, 2n-2, \ldots, n+3, 1, n+2, n, n-1, \ldots, 4, 2, 3)$. We can spot the two sole transpositions between these permutations, namely $(2n, 2n+1)$ and $(2, 3)$. The effects of such transpositions are:

- the tree edge whose tail is $2n$ goes to $2n+1$ in $G_1$ and to $2n+2$ in $G_2$;
- the tree edge whose tail is $2n-1$ goes to $2n$ in $G_1$ and to $2n+1$ in $G_2$;
- the tree edge whose tail is 2 goes to 3 in $G_1$ and to 4 in $G_2$;

Since all other tree edges remain the same, there are exactly three vertices whose parents in the representative tree are different in $G_1$ and $G_2$, namely $2n, 2n-1$ and 2.

We remark that there are many such pairs $\omega_1, \omega_2$ for each value of $n$. The one above is but an example. Indeed, computational results suggest that the ratio of canonical reducible permutation graphs encoding binaries of $n$ bits which can be modified into another canonical reducible permutation graph of the same size by replacing three edges grows with $n$, as discussed in the concluding Section 6. $\square$

## 5. Proof of Theorem 10

Let $\mathcal{G}_k$ be the set of all canonical reducible permutation graphs with $k$ edges missing. When an element $G'$ of $\mathcal{G}_k$ is the input of *plug_next_subpath* $(G', \emptyset, \mathcal{H})$, its output is clearly a Hamiltonian path of some graph $G$ such that $V(G) = V(G')$ and $E(G) \setminus E(G') \leq k$. Thus, when a canonical reducible permutation $G$ minus two edges is passed to Algorithm 1, the path $H$ it returns is the Hamiltonian path of some element of $\mathcal{G}_2$. We claim such graph can be no other but $G$.

Let $\hat{H} = 2n+2, 2n+1, \ldots, 0$ be the unique Hamiltonian path of $G$. We divide the proof in three cases: (i) the removed edges were both tree edges of $G$; (ii) the removed edges were both path edges of $G$; (iii) the removed edges were one tree edge and one path edge of $G$.

*When two tree edges are missing.* The easiest case is (i), as illustrated in Figure 2(a). If only tree edges were removed, then the Hamiltonian path of $G$ is undamaged. Starting from the only vertex with out-degree zero in $G'$, namely vertex 0, *plug_next_subpath*$(G', \emptyset, \mathcal{H})$ outputs $\hat{H}$ at once, never making a single recursive call. Since $\hat{H}$ obviously produces the correct labeling of vertices of $G$, it is validated uneventfully by *validate_labels*$(G', \hat{H})$ and returned by the algorithm.

*When two path edges are missing.* Suppose now that (ii) holds. Since no tree edges were removed, the only vertex with out-degree zero in $G'$ is vertex 0, unless the path edge whose tail is $2n+2$ was one of the removed edges (we recall that $2n+2$ has degree 1 in $G$). We therefore analyze two subcases, according to whether or not $(2n+2, 2n+1)$ was removed from $E(G)$.

For the first subcase, suppose the removed path edges were $(2n+2, 2n+1)$ and $(a, a-1)$ for some integer $a$ with $1 \leq a \leq 2n+1$, as in Figure 2(b). Although vertex $2n+2$ has degree zero, its in-degree is greater than 1 in $G' \setminus \emptyset = G'$, and therefore no $\{2n+2\}$-bup is produced in the main call to *plug_next_subpath*. Thus, the only partial path the algorithm produces, starting from vertex 0 in backwards fashion, is $Q' = a-1, a-2, \ldots, 0$. Because now $a-1$ has no in-neighbors in $G - V(Q')$, it recurses to find possible extensions for $Q'$. The only vertices with degree 0 or 1 in $G' \setminus V(Q')$ are now $2n+2$ and $a$. However, any $\{2n+2\}$-bup $Q''$ that may be found will constitute an $H = Q''||Q$ path that will necessarily fail the ensuing validation. This is due to the fact that $H[2n+2]$ will be a vertex other than $\hat{H}[2n+2] = 2n+2$, an out-neighbor of $n+1$ by Theorem 9—and

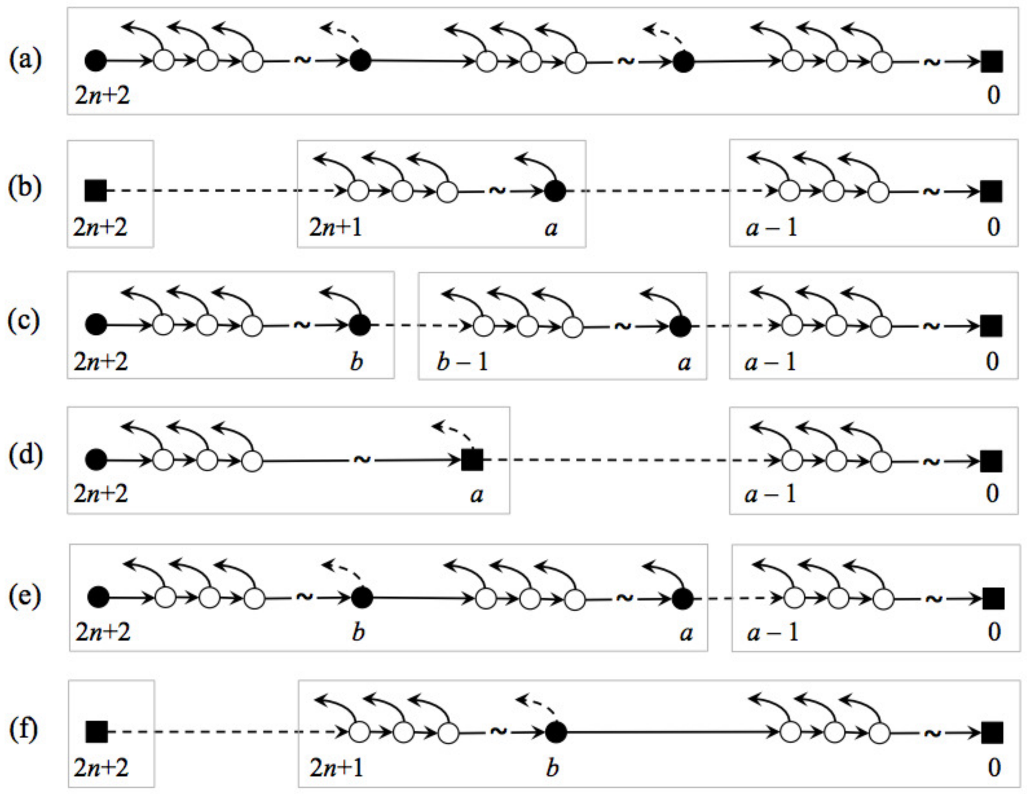

Figure 2: Possible scenarios for the Hamiltonian path $H$ of a damaged watermark $G'$. Dashed arrows indicate missing edges. Squares, solid circles and hollow circles represent vertices whose out-degrees in $G'$ are, respectively, 0, 1 and 2. Three hollow circles close together followed by a broken arrow (with a tilde in the middle) indicate subpaths of zero or more edges. Each big rectangle encloses a maximal undamaged subpath of $H$, which corresponds to a maximal backward-unbifurcated path to $s \in V(G)$, i.e., an $\{s\}$-bup.

the first condition tested by *validate_labels*$(G', H)$ cannot be met. Because of this, the partial path $Q'$ can only be extended by an $\{a\}$-bup, which can be no other but $2n+1, \ldots, a$, and the remaining vertex $2n+2$ will be concatenated during the next recursive call, completing the Hamiltonian path $2n+2, 2n+1, \ldots, a, a-1, \ldots, 0 = \hat{H}$, as desired.

In the second subcase, the path edge $(2n+2, 2n+1)$ was not removed. Suppose the missing edges are $(a, a-1)$ and $(b, b-1)$, with $a < b$, as illustrated in Figure 2(c). The first, rightmost subpath located by the algorithm can only be the unique $\{0\}$-bup, namely $Q' = a-1, a-2, \ldots, 0$. Now there are three vertices whose degree are less than or equal to one: $2n+2, b$ and $a$.

When the algorithm considers $\{2n+2\}$-bups during the recursive call to *plug_next_subpath*$(G' - V(Q'), Q'||\emptyset, \mathcal{H})$, whichever ensuing Hamiltonian path candidate $H$ it produces will necessarily be discarded. Indeed, if $n+1 \geq a$, then no bup is even produced because the in-degree of $2n+2$ in $G' - V(Q')$ is at least 2 by the existence of tree edges $(n+1, 2n+2)$ and $(2n+1, 2n+2)$; and, if $n+1 < a$, then $H[n+1]$ is vertex $\hat{H}[n+1] = n+1$ itself and, because its out-degree in $G'$ is already 2, conditions (1) and (2) checked by *validate_labels*$(G', H)$ cannot both be met.

When the algorithm considers a $\{b\}$-bup, whichever ensuing Hamiltonian path candidate $H$ it comes up with will also be discarded. Indeed, because the subpath $2n+2, 2n+1, \ldots, b$ of $\hat{H}$ is intact, vertex $2n+2$ will be brought into the $\{b\}$-bup before $v$ does, for all $b-1 \geq v \geq a$, hence $H[2n+2] \neq 2n+2$. Moreover, because in particular the tree edge whose tail is $b-1$, say $(b-1, w)$, was not removed, and $w \geq b$, the only possible value for $w$ is $2n+2$, otherwise there would be a vertex $z \in \{2n+1, 2n, \ldots, b\}$ with in-degree greater than 1 in the subgraph of $G'$ induced by $z$ and by the vertices to the left of $z$ in $H$, which is a contradiction because such path would have been discarded in the last line of Algorithm 2. Thus, vertex $b-1$ is an in-neighbor of $2n+2$ which was not added to the path before $2n+2$ was added. If a backward bifurcation has not arisen, then it is only possible that $b-1$ is precisely the vertex to the left of $2n+2$ in $H$. Repeating the same argument—based on the fact that the tree edge whose tail is $v$ has not been removed—for all $b-2 \geq v \geq a$, we can infer that the only possible Hamiltonian path candidate produced by the concatenation of a $\{b\}$-bup to the left of $Q'$ is $H = a, a+1, \ldots, b-1, 2n+2, 2n+1, \ldots, b, a-1, a-2, \ldots 0$. Now, condition (1) in *validate_labels* enforces that $H[n+1]$ is the tail of a tree edge pointing to $H[2n+2] = a$. However, because $H[n+1] > a$, such edge cannot be an actual tree edge of the original graph $G$, hence it must be a path edge. Since the only path edge with head $a$ in $G$ is $a+1$, it follows that $H[n+1] = a+1$. And here we shall have a contradiction, since $a+1$ is the second vertex, left to right, in $H$ (i.e., $H[2n+1] = n+1$), unless $a = b-1$. However, if $a = b-1$, then $H[n+1] = b$, and the existence of edge $(b, a) = (H[n+1], H[2n+2])$ is necessary to meet condition (1) in the validation procedure. But $(b, a) = (b, b-1)$ is one of the removed edges, therefore it must be reinserted. Condition (2), on its turn, requires that an outgoing edge is added to $2n+2$ (whose degree is 1 and whose index $i$ in $H$ satisfies $2n+1 \geq i \geq 1$). Along with the plausible path edge $(b, a-1)$, which was required to concatenate the $\{b\}$-bup to the left of $Q'$, we have a total of

3 new edges, thus violating condition (3).

Finally, when the algorithm considers $\{a\}$-bups, it necessarily produces the subpath $Q'' = b-1, \ldots, a$, which is concatenated to $Q'$, and, because $\{2n+2\}$-bups cannot possibly yield a valid prefix to $Q''||Q'$, the last recursive call can only produce the $\{b\}$-bup $2n+2, \ldots, b$, reconstituting $\hat{H}$.

*When a tree edge and a path edge are missing.* We focus on the final case (iii), where one path edge and one tree edge were removed. We now consider three subcases separately. In the first one, both the path edge and the tree edge that were removed share the same tail endpoint. In the second one, the tails of the removed edges are distinct. The third case is a special case of the second one, when the tail of the removed path edge is $2n+2$.

For the first subcase, illustrated in Figure 2(d), say both removed edges have tail $a \in V(G')$. In this case, vertex $a$ presents degree zero, just like vertex 0 itself. Any attempts to build a Hamiltonian path $H$ whose suffix is an $\{a\}$-bup, however, shall not succeed. Since vertex $2n+2$ will be brought into $H$ before vertex 0 does, and because $a$ will be the rightmost vertex in $H$ (i.e., $H[0] = a$), a tree edge leaving $2n+2$ is necessary to satisfy condition (2) of *validate_labels*$(G', H)$. But vertex 0 appears with index $i > 0$ in $H$, and therefore a plausible path edge must be inserted with 0 as its tail. If the index of 0 is not $2n+2$, then a tree edge leaving 0 is also called for. If, on the other hand, the index of 0 is $2n+2$, then, among the two tree edges reaching $H[2n+2] = 0$ that are required by condition (1) of the validation procedure, at least one of them is still missing. In both cases, condition (3) is violated.

The second case is the one depicted in Figure 2(e), where a path edge $(a, a-1)$, with $1 < a \leq 2n+1$, and a tree edge $(b, v)$, with $v > b$, were removed. Procedure *plug_next_subpath* starts by gathering the maximal backward-unbifurcated path $Q'$ whose head is 0, the only vertex with degree zero in $G'$. The leftmost vertex of such $\{0\}$-bup is vertex $a-1$, the first vertex whose in-degree is zero in the subgraph of $G'$ induced by vertices not in $Q'$, and hence $Q' = a-1, a-2, \ldots, 0$. Now three vertices have out-degree less than or equal to one: $2n+2$, $b$ and $a$.

When the algorithm picks $2n+2$ as a possible continuation of the backward path under construction, the index of $2n+2$ in $H$ will be $a$. By Theorem 9, vertex $2n+1$ is always a child of the root $2n+2$ in the representative tree $T$ of a canonical reducible permutation graph $G$, and, by Property 2, the number of children $v \leq 2n$ of $2n+2$ in $T$ corresponds to the number $n_1$ of digits 1 in the binary representation $B$ of the identifier $\omega$

encoded by $G$. As a consequence, the in-degree of $2n+2$ in $G$ is $n_1+1$. We now tackle two distinct situations. In the first one, $a \le n+1$, whereas in the second one $a > n+1$. If $a \le n+1$, then the in-degree of $2n+2$ in $G' - V(Q')$ is the same as in $G'$ (i.e., $n_1$), since all in-neighbors of $2n+2$ belong to $\{n+1, \ldots, 2n+1\}$ by the same Theorem 9. Because, along with the path edge $(a, a-1)$, only one tree edge was removed from $G$ to obtain $G'$, the in-degree of $2n+2$ in $G'$ is at least $n_1+1-1 = n_1$. As a consequence, a backward bifurcation would be noticed on $2n+2$ unless $n_1 = 1$ *and* the tail $b$ of the removed tree edge is one of the in-neighbors of $2n+2$, which in this case are $n+1$ and $2n+1$. If $b = n+1$, then the tree edge $e = (2n+1, 2n+2)$ is intact, and the only possible placement of vertex $2n+1$ in $H$ is at the position immediately to the left of $2n+2$, so that $e$ functions as a path edge of $H$. Assuming there was no backward bifurcation on $2n+2$ (which would have caused the path $H$ to be discarded), the only possible tree edge leaving $2n$ is $(2n, 2n+1)$, hence $2n$ must be placed to the left of $2n+1$ in $H$. Assuming, similarly, that no backward bifurcation occurred on $2n+1$, the only possible tree edge leaving $2n-1$ is $(2n-1, 2n)$, and so on. This reasoning must continue until finally $a$ is concatenated at the very first position of $H$, yielding $H = a, a+1, \ldots, 2n+2, a-1, a-2, \ldots, 0$. Now, condition (1) of the validation procedure requires that $H[n+1]$ and $H[2n+1]$ are in-neighbors of $H[2n+2] = a$. However, this requirement and condition (2) cannot both be met without violating condition (3), because, since those two vertices $H[n+1]$ and $H[2n+1]$ are not in $Q'$, they are certainly greater than $a$, but there is only one vertex in $G$ which is greater than $a$ and is an in-neighbor of $a$, namely $a+1$. Therefore an extra tree edge is required, but one extra edge is also required by condition (2)—a tree edge leaving $b$—and the plausible path edge $(2n+2, a-1)$ had already been inserted, which breaks condition (3). We are left with the possibility that the tail of the removed tree edge was $b = 2n+1$. In this case, the tree edge $(n+1, 2n+2)$ is intact, and the vertex immediately to the left of $2n+2$ in $H$ must be $n+1$. Now, since path edge $(n+2, n+1)$ is not the missing one by hypothesis, vertex $n+2$ must be immediately to the left of $n+1$ in $H$, and, since path edge $(n+3, n+2)$ is not the missing one, vertex $n+3$ must appear immediately to the left of $n+2$, and so on, until $b = 2n+1$ is concatenated at the first position of $H$, yielding $H = 2n+1, 2n, \ldots, a, 2n+2, a-1, a-2, \ldots, 0$. To satisfy condition (1) of the validation, vertex $H[n+1]$ must be an in-neighbor of $H[2n+2] = 2n+1$. But, because $n_1 = 1$ ($\omega$ is a power of 2), the root of its Type-2 representative tree has only two children, which allows item (iii) in the definition of Type-2 trees (Definition 7) to assure that $2n+1$ has only one child, and this child is not $n+1$, by item (i) of that same definition. Thus, the tree edge $(H[n+1], 2n+1)$ must be added to satisfy condition (1) of *validate_labels*$(G', H)$, and the only vertices with out-degree 1 in $G'$ were $b$, which is $2n+1$ itself, $2n+2$, which was already added a plausible path edge connecting it to $a-1$, and $a$. It is therefore only possible that $H[n+1] = a$, that is, the missing path edge is necessarily $(n+1, n)$. And here is where condition (4) of the validation procedure comes into play, enforcing that the root $H[2n+2]$ presents only two children when $\omega$ is a power of 2. Since that is not the case for the path $H$ so obtained, as can be easily checked, $H$ is discarded. The second situation is the one in which $a > n+1$. This one is easy, since now $H[n+1] = n+1$, which is the tail of a tree edge pointing to $2n+2 \neq H[2n+2]$, and hence conditions (1) and (2) of the validation cannot both be met, unless such tree edge is precisely the one tree edge that was removed. But that would correspond to the subcase shown in Figure 2(d), which we already tackled.

When the algorithm picks $b$ as the head of the first subpath to extend the $\{0\}$-bup $Q'$, all ensuing Hamiltonian path candidates shall be discarded by similar reasons. Finally, when it considers the sound continuation $a$, all conditions obviously pass and $\hat{H}$ is delivered.

The third—and last—possible situation is the one depicted in Figure 2(f), where the removed edges were the path edge $(2n+2, 2n+1)$ and a tree edge $(b, v)$, with $v > b$. There are two vertices with degree zero: 0 and $2n+2$. When the call to *plug_next_subpath*$(G', \emptyset, \mathcal{H}\})$ picks $2n+2$ as the rightmost vertex of $Q'$, the leftmost vertex of whatever Hamiltonian path $H$ it produces must be either 0 or $b$, the only vertices with out-degree less than 2 in $G'$ (part of the second condition verified by *validate_labels*). Moreover, the root of the representative tree of $G$ must have only two children (which means $n_1 = 1$, or, equivalently, the identifier $\omega$ encoded by $G$ is a power of 2), and $b$ must be either $2n+1$ or $n+1$, so that a backward bifurcation does not take place at the very starting vertex $H[0] = 2n+2$. If $H[2n+2] = 0$, then at least three extra edges are required to put $H$ together and satisfy condition (1) of the validation procedure: a plausible path edge $(H[2n{+}2], H[2n{+}1])$, and at least two tree edges, namely $(H[2n+1], H[2n+2])$ and $(H[n+1], H[2n+2])$. But then, of course, condition (3) is violated. If $H[2n+2] = b = n+1$, then the vertex immediately to the left of $H[0] = 2n+2$ in $H$ must be $H[1] = 2n+1$, and the next vertex right-to-left must be $H[2] = 2n$ and so on, assuming no backward bifurcations took place, until at least vertex $H[n] = n+2$. To

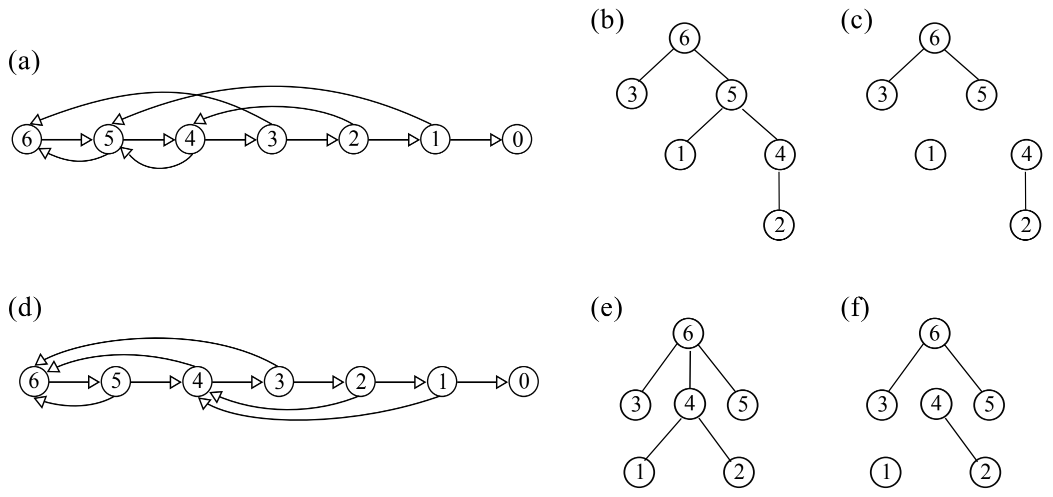

Figure 3: (a) The watermark $G_1$ for identifier $\omega = 2$; (b) its representative tree $T_1$; (c) the damaged representative tree $T_1' = T_1 - \{(1,5),(4,5)\}$; (d) the watermark $G_2$ for identifier $\omega = 3$; (e) its representative tree $T_2$; (f) the damaged representative tree $T_2' = T_2 - \{(1,4),(4,6)\}$. Note that $T_1'$ and $T_2'$ are isomorphic.

put it differently, the $\{2n+2\}$-bup $Q'$ considered initially by the algorithm contains (not necessarily properly, depending on whether there was a tree edge pointing to $n+2$ in $G'$) the suffix $Q' = n+2, n+3, \ldots 2n+2$. Now, no matter which vertex $w$ occupies the $(n+1)$th position (right-to-left) in $H$, it was certainly not an in-neighbor of $H[2n+2] = n+1$, because $n+1$ does not have in-neighbors in Type-2 trees (and in Type-1 trees neither, for that matter). If $w \neq 0$, then $w$ has out-degree 2, and conditions (1) and (2) of the validation procedure cannot both be met. If, on the other hand, $w = 0$, then $H$ is the concatenation of $Q'$ with the prefix $n+1, n, n-1, \ldots, 0$, an intact subpath of $\hat{H}$. In this case, vertex $H[2n+1]$ is $n$, a vertex with out-degree 2 in $G'$ which is not an in-neighbor of $H[2n+2] = n+1$, and conditions (1) and (2), again, cannot both be met.

The verification of the time complexity is straightforward. □

## 6. Final considerations

Relying on the characterization of canonical reducible permutation graphs given in [1], we were able to formulate a linear-time algorithm which succeeds in retrieving deterministically the $n$-bit identifiers encoded by such graphs

Table 1: The first columns show the ratio $r(n,k) = |\Omega(n,k)| \ / \ |\Omega(n)|$ between $|\Omega(n,k)|$, the number of canonical reducible permutation graphs encoding $n$-bit identifiers which can become isomorphic to another such graph by the removal of $k$ edges, and $|\Omega(n)|$, the total number of such graphs, for $3 \leq n \leq 15$ and $3 \leq k \leq 5$; the remaining columns show the probability $p(n,k)$ that a canonical reducible permutation graph becomes irremediably damaged by the removal of $k$ edges chosen uniformly at random, for those same $n$ and $k$.

| $n$ | $r(n,3)$ | $r(n,4)$ | $r(n,5)$ | $p(n,3)$ | $p(n,4)$ | $p(n,5)$ |
|---|---|---|---|---|---|---|
| 3 | 50.00% | 100.00% | 100.00% | 5.71429% | 8.57143% | 33.33333% |
| 4 | 25.00% | 100.00% | 100.00% | 2.38095% | 2.57937% | 5.09607% |
| 5 | 50.00% | 93.75% | 100.00% | 1.21212% | 1.93182% | 2.91899% |
| 6 | 68.75% | 90.62% | 100.00% | 0.69930% | 1.23876% | 1.92696% |
| 7 | 81.25% | 92.19% | 98.44% | 0.43956% | 0.76190% | 1.21360% |
| 8 | 89.06% | 94.53% | 97.66% | 0.29412% | 0.47731% | 0.75934% |
| 9 | 93.75% | 96.48% | 98.05% | 0.20640% | 0.30999% | 0.48427% |
| 10 | 96.48% | 97.85% | 98.63% | 0.15038% | 0.20897% | 0.31842% |
| 11 | 98.05% | 98.73% | 99.12% | 0.11293% | 0.14571% | 0.21637% |
| 12 | 98.93% | 99.27% | 99.46% | 0.08696% | 0.10463% | 0.15169% |
| 13 | 99.41% | 99.58% | 99.68% | 0.06838% | 0.07707% | 0.10939% |
| 14 | 99.68% | 99.77% | 99.82% | 0.05473% | 0.05803% | 0.08086% |
| 15 | 99.83% | 99.87% | 99.90% | 0.04449% | 0.04453% | 0.06108% |

(with $n > 2$) even if $k \leq 2$ edges are missing. Such bound is the best possible, since there are plenty of watermark instances whose edge sets differ in no more than three edges. That is obviously not the case for only two edges, otherwise our algorithm would not have been possible. Indeed, the sole example of two canonical reducible permutation graphs which may become isomorphic to one another when each graph is deprived of only two edges occurs when $n = 2$, as illustrated in Figure 3. An interesting open problem is to characterize the maximum sets $\Omega(n,k)$ of $n$-bit identifiers, such that, for all $\omega_1, \omega_2 \in \Omega(n,k)$, the corresponding watermarks $G_1, G_2$ satisfy

$$|E(G_1) \setminus E(G_2)| = |E(G_2) \setminus E(G_1)| = k.$$

For the time being, we know that $\Omega(2,2) = \{2,3\}$, and $\Omega(n,2) = \emptyset$ for $n > 2$. We also know, as pointed out in the proof of Corollary 15, that $\{2^{n-1}, 2^{n-1}+1\} \subseteq \Omega(n,3)$ for $n > 3$. Furthermore, computational results suggest that the ratio $|\Omega(n,k)| \ / \ |\Omega(n)|$—where $\Omega(n)$ is the set of all $2^n$ canonical reducible permutation graphs encoding $n$-bit identifiers—approaches 1 as $k$ grows, which is rather intuitive, and also approaches 1 as $n$ grows, which may be less intuitive (see Table 1).

## References

[1] L.M.S. Bento, D.R. Boccardo, R.C.S. Machado, V.G. Pereira de Sá, J.L. Szwarcfiter, Full characterization of a class of graphs tailored for software watermarking, submitted to *Journal of Computer and System Sciences*, 2014.

[2] L.M.S. Bento, D.R. Boccardo, R.C.S. Machado, V.G. Pereira de Sá, J.L. Szwarcfiter, Towards a provably resilient scheme for graph-based watermarking, *Proc. 39th International Workshop on Graph-Theoretic Concepts in Computer Science, WG'13*, LNCS **8165** (2013), 50–63, arXiv: 1302.7262v6 [cs.MM].

[3] M. Chroni, S.D. Nikolopoulos, Efficient encoding of watermark numbers as reducible permutation graphs, arXiv:1110.1194v1 [cs.DS], 2011.

[4] M. Chroni, S.D. Nikolopoulos, An efficient graph codec system for software watermarking, *Proc. 36th IEEE Conference on Computers, Software and Applications, COMPSAC'12*, IEEE Proc. (2012), 595–600.

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

[6] C. Collberg, C. Thomborson, G. Townsend, Dynamic graph-based software fingerprinting, *ACM Trans. Programming Languages and Systems* **29** (2007), 1–67.

[7] C. Collberg, A. Huntwork, E. Carter, G. Townsend, M. Stepp, More on graph theoretic software watermarks: implementation, analysis and attacks, *Information and Software Technology* **51** (2009), 56–67.

[8] R. L. Davidson, N. Myhrvold, Method and system for generating and auditing a signature for a computer program, US Patent 5.559.884, Microsoft Corporation (1996).

[9] M.S. Hecht and J.D. Ullman, Flow graph reducibility, *SIAM Journal of Computing* **1** (1972), 188–202.

[10] M.S. Hecht and J.D. Ullman, Characterizations of reducible flow graphs, *Journal of the ACM* **21** (1974), 367–375.

[11] R.E. Tarjan, Testing flow graph reducibiliy, *Journal of Computer and System Sciences* **9** (1974), 355–365.

[12] R. Venkatesan, V. Vazirani, Technique for producing through watermarking highly tamper-resistant executable code and resulting watermarked code so formed (2006), Microsoft Corporation, US Patent: 7051208.