Towards a provably resilient scheme for graph-based watermarking

Lucila Maria Souza Bento
Davidson Boccardo
Raphael Carlos Santos Machado
Vinícius Gusmão Pereira de Sá
Jayme Luiz Szwarcfiter
Watermarks
Watermarks
Watermarks
Software watermarking

What for?

How?
Software watermarking

```cpp
int fibonacci (int n) {
    int a = 1, b = 1;

    for (int i = 1; i < n; i++) {
        int sum = a + b;
        a = b;
        b = sum;
    }

    return b;
}
```
int fibonaci (int n) {
    int a = 1, b = 1;
    // author: Vinícius
    for (int i = 1; i < n; i++) {
        int sum = a + b;
        a = b;
        b = sum;
    }
    return b;
}
Software watermarking

```c
int fibonnaci (int n) {
    int a = 1, b = 1;
    string author = "Vinícius";
    for (int i = 1; i < n; i++) {
        int sum = a + b;
        a = b;
        b = sum;
    }

    return b;
}
```
Graph-based software watermarking

int fibonacci (int n) {
    int a = 1, b = 1;

    for (int i = 1; i < n; i++) {
        int sum = a + b;
        a = b;
        b = sum;
    }

    return b;
}
Graph-based software watermarking

Davidson and Myhrvold (1996)
Venkatesan, Vazirani and Sinha (2001)
Collberg et al. (WG 2003)

“author: Vinícius”

```c
int fibonacci (int n) {
    int a = 1, b = 1;

    for (int i = 1; i < n; i++) {
        int sum = a + b;
        a = b;
        b = sum;
    }

    return b;
}
```
Graph-based software watermarking

```c
int fibonacci(int n) {
    int a = 1, b = 1;

    for (int i = 1; i < n; i++) {
        int sum = a + b;
        a = b;
        b = sum;
    }

    return b;
}
```

“author: Vinícius”

(100101010001110101001101011)
Graph-based software watermarking

```c
int fibonacci (int n) {
    int a = 1, b = 1;
    for (int i = 1; i < n; i++) {
        int sum = a + b;
        a = b;
        b = sum;
    }
    return b;
}
```

“author: Vinícius”

(100101010001110101001101011)

Davidson and Myhrvold (1996)
Venkatesan, Vazirani and Sinha (2001)
Collberg et al. (WG 2003)

Control flow graph
Graph-based software watermarking

```c
int fibonnaci (int n) {
    int a = 1, b = 1;

    for (int i = 1; i < n; i++) {
        int sum = a + b;
        a = b;
        b = sum;
    }

    return b;
}
```

“author: Vinícius”

(100101010001110101001101011)

Control flow graph
Graph-based software watermarking

int fibonacci (int n) {
    int a = 1, b = 1;

    for (int i = 1; i < n; i++) {
        int sum = a + b;
        a = b;
        b = sum;
    }

    return b;
}
Graph-based software watermarking

Encoding

Decoding

10010101000111010101001101011

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key $\omega = 29$
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

$\bar{B} = 00010$

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

$\bar{B} = 00010$

$B^* = 11111\ 00010\ 0$

$n$ 1’s
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101$  \quad n = 5$

$\bar{B} = 00010$

$B^* = 11111 \, 00010 \, 0$

$n$ 1’s

$Z_0 = 6, 7, 8, 10, 11$

$Z_1 = 1, 2, 3, 4, 5, 9$

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

$\bar{B} = 00010$

$B^* = 11111 \ 00010 \ 0$

$n$ 1’s

$Z_0 = 6, 7, 8, 10, 11$

$Z_1 = 1, 2, 3, 4, 5, 9$

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

Chromni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

$\overline{B} = 00010$

$B^* = 11111 \ 00010 \ 0$

$n \ 1\text{'s}$

$Z_0 = 6, 7, 8, 10, 11$

$Z_1 = 1, 2, 3, 4, 5, 9$
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

$\bar{B} = 00010$

$B^* = 11111 \ 00010 \ 0$

$n$ 1’s

$Z_0 = 6, 7, 8, 10, 11$

$Z_1 = 1, 2, 3, 4, 5, 9$

Chroni and Nikolopoulos (2011)

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

$\overline{B} = 00010$

$B^* = 11111 00010 0$

$n$ 1's

$Z_0 = 6, 7, 8, 10, 11$

$Z_1 = 1, 2, 3, 4, 5, 9$

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 4, 5$

fixed element

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101$  \quad n = 5$

$\bar{B} = 00010$

$B^* = 11111\ 00010\ 0$

$n$ 1's

$Z_0 = 6, 7, 8, 10, 11$

$Z_1 = 1, 2, 3, 4, 5, 9$

Chroni and Nikolopoulos (2011)

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

$\bar{B} = 00010$

$B^* = 11111 \ 00010 \ 0$

$n \ 1's$

$Z_0 = 6, 7, 8, 10, 11$

$Z_1 = 1, 2, 3, 4, 5, 9$

Chroni and Nikolopoulos (2011)

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

$\ Z_0 \ 
\ Z_1^R \ 

6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$
The codec from Chroni and Nikolopoulos:

- Key: $\omega = 29$
- $B = 11101$, $n = 5$
- $\bar{B} = 00010$
- $B^* = 11111000100$ ($n$ 1's)

- $Z_0 = 6, 7, 8, 10, 11$
- $Z_1 = 1, 2, 3, 4, 5, 9$

- $P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$
- $P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$
- $12, 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

- $2n+2$
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101$ \hspace{0.5cm} $n = 5$

$\bar{B} = 00010$

$B^* = 11111 \ 00010 \ 0$

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

$12, 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

$Z_0$

$Z_1^R$

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101$ \hspace{1cm} $n = 5$

$\bar{B} = 00010$

$B^* = 11111 \ 00010 \ 0$

$n$ 1’s

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

$12, 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key \( \omega = 29 \)

\[ B = 11101 \quad n = 5 \]

\[ \bar{B} = 00010 \]

\[ B^* = 11111 \quad 00010 \quad 0 \]

\[ P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1 \]

\[ P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5 \]

\[ 12, 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5 \]

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

$\bar{B} = 00010$

$B^* = 11111 \ 00010 \ 0$

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

$12, 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

$Z_0$

$Z_1^R$

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = \overline{B} = 11101$
$n = 5$

$\bar{B} = 00010$

$B^* = 11111\ 00010\ 0$

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

$12, 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$
The codec from Chroni and Nikolopoulos

key \( \omega = 29 \)

\( B = 11101 \quad n = 5 \)

\( \bar{B} = 00010 \)

\( B^* = 11111 \quad 00010 \quad 0 \)

\( P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1 \)

\( P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5 \)

\( 12, 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5 \)
The codec from Chroni and Nikolopoulos

key \( \omega = 29 \)

\[ B = 11101 \quad n = 5 \]

\[ \bar{B} = 00010 \]

\[ B^* = 11111 \ 00010 \ 0 \]

\[ n \text{ 1's} \]

\[ P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1 \]

\[ P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5 \]

\[ 12, 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5 \]

Chroni and Nikolopoulos (2011)
The codec from Chroni and Nikolopoulos
The codec from Chroni and Nikolopoulos

key $\omega = 29$

$B = 11101 \quad n = 5$

$\bar{B} = 00010$

$B^* = 11111\ 00010\ 0$

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

$12, 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

Chroni and Nikolopoulos (2011)
Our contribution

1. **full characterization of the class of canonical reducible permutation graphs** (the graphs produced by Chroni and Nikolopoulos’s encoding algorithm)

2. a **linear-time recognition algorithm** for such graphs

3. a **new linear-time decoding algorithm** (graph $\rightarrow$ integer key) simpler, marginally faster and able to retrieve the correct key even after the malicious removal of $k \leq 2$ edges

4. a **tight bound for the resilience of the codec against edge removals**
Our contribution

1. **formal definition of the class** of *canonical reducible permutation graphs* (precisely the graphs produced by Chroni and Nikolopoulos’s’s encoding algorithm)

2. characterization and linear-time recognition algorithm for such graphs

3. a new linear-time decoding algorithm (graph $\rightarrow$ integer key) simpler, marginally faster and able to retrieve the correct key even after the malicious removal of $k \leq 2$ edges

4. a tight bound for the resilience of the codec against edge removals

B., B., M., S., S. (WG 2013)
Canonical reducible permutation graphs
Canonical reducible permutation graphs

canonical reducible permutation

self-labeling reducible flow graph
**Definition**

*Self-labeling reducible flow graph* $G(V,E)$:
- vertices $0, \ldots, |V|-1$
- exactly one Hamiltonian path
- $v$ in $V \setminus \{0, |V|-1\} \implies N^+(v) = \{v-1, w\}$, for some $w > v$
- $v = 0 \implies N^+(v) = \{\}$
- $N^-(v) = \{1\}$
- $v = |V|-1 \implies N^+(v) = \{|V|-2\}$
- $|N^-(v)| \geq 2$
Canonical reducible permutation graphs
Canonical reducible permutation graphs
Canonical reducible permutation graphs
Canonical reducible permutation graphs

Representative tree

- children in ascending order
- max-heap property
Canonical reducible permutation graphs

Representative tree

- children in ascending order
- max-heap property

root-free preorder traversal:
6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5
Definition

Canonical self-inverting permutation

- a self-inverting permutation
- elements $s_i = 1, 2, \ldots, 2n+1$
- exactly one fixed element
- each 2-cycle $(s_i, s_j)$ satisfies
  \[ 1 \leq i \leq n, \quad s_i > s_j \]
Definition

Canonical reducible permutation graph:
- a self-labeling reducible flow graph
- $2n+3$ vertices
- its representative tree has a (root-free) preorder traversal which is a canonical self-inverting permutation
Theorem

*Watermark from Chroni and Nikolopoulos*

*Canonical reducible permutation graph*
Canonical reducible permutation graphs

Theorem

Watermark from Chroni and Nikolopoulos

Canonical reducible permutation graph

(Proof by “we don’t want to know the details” argument)
Canonical reducible permutation graphs

key ω = 29

$B = 11101 \quad n = 5$

$\bar{B} = 00010$

$B^* = 11111 \ 00010 \ 0$

$P_b = 6, 7, 8, 10, 11, 9, 5, 4, 3, 2, 1$

$P_s = 6, 7, 8, 10, 11, 1, 2, 3, 9, 4, 5$

$Z_0$

$Z_1^R$

Chroni and Nikolopoulos (2011)

several structural properties
Codec properties

Property 1 For $1 \leq i \leq n$, element $b_{n+i+1}$ in $P_b$ is equal to $n - i + 1$, that is, the $n$ rightmost elements in $P_b$ are $1, 2, \ldots, n$ when read from right to left.

Property 2 The elements whose indexes are $1, 2, \ldots, n$ in $P_s$ are all greater than $n$.

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

Property 4 In self-inverting permutation $P_s$, elements indexed $1, 2, \ldots, f - n - 1$ are respectively equal to $n + 1, n + 2, \ldots, f - 1$, and elements indexed $n + 1, n + 2, \ldots, f - 1$ are respectively equal to $1, 2, \ldots, f - n - 1$.

Property 5 The first element in $P_s$ is $s_1 = n + 1$, and the central element in $P_s$ is $s_{n+1} = 1$.

Property 6 If $f \neq n^*$, then the index of element $n^*$ in $P_s$ is equal to $n_1 + 1$, and vice-versa. If $f = n^*$, then the index of element $n^*$ in $P_s$ is also $n^*$.

Property 7 The subsequence of $P_s$ consisting of elements indexed $1, 2, \ldots, n + 1$ is bitonic.

Property 8 For $u \neq 2n + 1$, $(u, 2n + 2)$ 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 key $\omega$ represented by $G$.

Property 9 If $(u, k)$ is a tree edge of watermark $G$, with $k \neq 2n + 2$, then (i) element $k$ precedes $u$ in $P_s$; and (ii) if $v$ is located somewhere between $k$ and $u$ in $P_s$, then $v < u$. 
Canonical reducible permutation graphs

Theorem
Watermark from Chroni and Nikolopoulos

Canonical reducible permutation graph

(Proof by “we don’t want to know the details” argument)
Our contribution

1. **formal definition of the class** of *canonical reducible permutation graphs* (precisely the graphs produced by Chroni and Nikolopoulos’s encoding algorithm)

2. **characterization and linear-time recognition algorithm** for such graphs

3. a new linear-time decoding algorithm (graph $\rightarrow$ integer key) simpler, marginally faster and able to retrieve the correct key even after the malicious removal of $k \leq 2$ edges

4. a **tight bound for the resilience of the codec** against edge removals
Characterizing the watermark graphs
(canonical reducible permutation graphs)

Theorem (characterization)
Canonical reducible permutation graph

Self-labeling reducible flow graph such that:
• its fixed element is $2n+1$, and
  its representative tree is a “type-1” tree
or
• its fixed element belongs to $[n+2, 2n]$, and
  its representative tree is a “type-2” tree
Theorem (characterization)

Canonical reducible permutation graph

Self-labeling reducible flow graph such that:
- its fixed element is $2n+1$, and
  its representative tree is a “type-1” tree
or
- its fixed element belongs to $[n+2, 2n]$, and
  its representative tree is a “type-2” tree
Types of representative trees

**type-1**

(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 \(2n\).

**type-2**

(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 \(N_T^*[f]\).

(f denotes the unique fixed element)
Linear-time recognition

Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph.
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph... provided we have the vertex labels!
Linear-time recognition

Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph.... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Linear-time recognition

Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph… provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph.... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Linear-time recognition

Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Linear-time recognition

Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph.... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph... *provided we have the vertex labels!*

Linear-time algorithm to find the unique Hamiltonian path
Linear-time recognition

Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph.... *provided we have the vertex labels!*

**Linear-time algorithm to find the unique Hamiltonian path**
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph.... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph.... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph... provided we have the vertex labels!

Linear-time algorithm to find the unique Hamiltonian path
Due to the characterization theorem, it is an easy task to recognize a canonical reducible permutation graph... *provided we have the vertex labels!*

*Linear-time algorithm to find the unique Hamiltonian path*
Linear-time recognition

Procedure 1: Reconstructing the Hamiltonian path

\[ V_0 \leftarrow \{ v \in V(G') \text{ s.t. } |N_{G'}^+| = 0 \}; \quad V_1 \leftarrow \{ v \in V(G') \text{ s.t. } |N_{G'}^+| = 1 \} \]

if \( |V_0| = 1 \) then
  let \( v_0 \) be the unique element in \( V_0 \)
  if \( |H(v_0)| = 2n + 3 \) then \( H \leftarrow H(v_0) \), return \( H \)
  else if \( \exists v_1 \in V_1 \) such that \( |H(v_0)| + |H(v_1)| = 2n + 3 \) then
    \( H \leftarrow H(v_1)||H(v_0) \), return \( H \)
else
  let \( v_1, v'_1 \in V_1 \) be such that
  \[ |H(v_0)| + |H(v_1)| + |H(v'_1)| = 2n + 3 \text{ and } N_{G'}^+(\text{first}(H(v_1))) \cap H(v'_1) \neq \emptyset \]
  \( H \leftarrow H(v'_1)||H(v_1)||H(v_0) \), return \( H \)
else
  let \( v_0, v'_0 \) be the elements in \( V_0 \)
  if \( |H(v_0)| + |H(v'_0)| = 2n + 3 \) then
    let \( v_0 \) be such that \( N_{G'}^+(\text{first}(H(v_0))) \cap H(v'_0) \neq \emptyset \)
    \( H \leftarrow H(v'_0)||H(v_0) \), return \( H \)
  else
    let \( v'_0 \in V_0 \) and \( v_1 \in V_1 \) be such that \( v'_0 \in N_{G'}^+(\text{first}(H(v_1))) \)
    \( H \leftarrow H(v'_0)||H(v_1)||H(v_0) \), return \( H \)
Linear-time recognition

Procedure 1: Reconstructing the Hamiltonian path

\[ V_0 \leftarrow \{ v \in V(G') \text{ s.t. } |N^+_G| = 0 \}; \quad V_1 \leftarrow \{ v \in V(G') \text{ s.t. } |N^+_G| = 1 \} \]

if \(|V_0| = 1\) then

let \(v_0\) be the unique element in \(V_0\)

if \(|H(v_0)| = 2n + 3\) then \(H \leftarrow H(v_0)\), return \(H\)

else if \(\exists v_1 \in V_1\) such that \(|H(v_0)| + |H(v_1)| = 2n + 3\) then

\(H \leftarrow H(v_1) || H(v_0)\), return \(H\)

else

let \(v_1, v'_1 \in V_1\) be such that

\(|H(v_0)| + |H(v_1)| + |H(v'_1)| = 2n + 3\) and \(N^+_G(first(H(v_1)) \cap H(v'_1)) \neq \emptyset\)

\(H \leftarrow H(v'_1) || H(v_1) || H(v_0)\), return \(H\)

else

let \(v_0, v'_0 \in V_0\)

if \(|H(v_0)| + |H(v'_0)| = 2n + 3\) then

let \(v_0\) be such that \(N^+_G(first(H(v_0))) \cap H(v'_0) \neq \emptyset\)

\(H \leftarrow H(v'_0) || H(v_0)\), return \(H\)

else

let \(v'_0 \in V_0\) and \(v_1 \in V_1\) be such that \(v'_0 \in N^+_G(first(H(v_1)))\)

\(H \leftarrow H(v'_0) || H(v_1) || H(v_0)\), return \(H\)
Our contribution

1. **formal definition of the class** of *canonical reducible permutation graphs* (precisely the graphs produced by Chroni and Nikolopoulos’s encoding algorithm)

2. **characterization** and **linear-time recognition algorithm** for such graphs

3. a **new linear-time decoding algorithm** (graph → integer key)
   
simpler, marginally faster and able to retrieve the correct key even after the malicious removal of \( k \leq 2 \) edges

4. a **tight bound** for the resilience of the codec against edge removals
A new decoding algorithm

1. find the unique Hamiltonian path and label the vertices accordingly
2. find the fixed element $f$
3. find the set $A$ of the child nodes of the root of the representative tree that are different from $2n+1$
4. calculate the key as follows

$$A = \begin{array}{c}
6 \\
7 \\
8 \\
10 \\
11 = 2n+1
\end{array}$$

$$\omega = \sum_{x_i \in A} 2^{2n-x_i}$$

Representative tree
A new decoding algorithm

1. find the unique Hamiltonian path and label the vertices accordingly
2. find the fixed element $f$
3. find the set $A$ of the child nodes of the root of the representative tree that are different from $2n+1$
4. calculate the key as follows

$$\omega = \sum_{x_i \in A} 2^{2n-x_i}$$
A new decoding algorithm

1. find the unique Hamiltonian path and label the vertices accordingly
2. find the fixed element $f$
3. find the set $A$ of the child nodes of the root of the representative tree that are different from $2n+1$
4. calculate the key as follows

with $k \leq 2$ missing edges
A new decoding algorithm

1. find the unique Hamiltonian path and label the vertices accordingly
2. find the fixed element $f$
3. find the set $A$ of the child nodes of the root of the representative tree that are different from $2n+1$
4. calculate the key as follows

Procedure 2: Finding $f \neq 2n + 1$

1. If $F$ contains a large vertex $x$ having a sibling $z$
   then let $f \leftarrow \max\{x, z\}$ and terminate the algorithm. Otherwise,
2. For each large vertex $x$ of $F$ satisfying $N_F(x) \neq \emptyset$ and each small $y \in N_F(x)$,
   let $Y' = \{x - n, x - n + 1, \ldots, n\}$. If $N_F^+(x) = Y'$ or $N_F^+(x) \subset Y'$,
   and $Y' \setminus N_F^+(x)$ is the vertex set of one of the trees of $F$,
   then let $f \leftarrow x$ and terminate the algorithm. Otherwise,
3. Find the preorder traversals of the three trees of $F$, and
   then let $f$ be the unique vertex that is both large and the rightmost element
   of the preorder traversal of some tree of $F$. 

A new decoding algorithm

1. find the unique Hamiltonian path and label the vertices accordingly
2. find the fixed element $f$
3. find the set $A$ of the child nodes of the root of the representative tree that are different from $2n+1$
4. calculate the key as follows

Procedure 2: Finding $f \neq 2n + 1$

1. If $F$ contains a large vertex $x$ having a sibling $z$ then let $f \leftarrow \max\{x, z\}$ and terminate the algorithm. Otherwise,
2. For each large vertex $x$ of $F$ satisfying $N_F(x) \neq \emptyset$ and each small $y \in N_F(x)$, let $Y' = \{x - n, x - n + 1, \ldots, n\}$. If $N_F^+(x) = Y'$ or $N_F^+(x) \subset Y'$, and $Y' \setminus N_F^+(x)$ is the vertex set of one of the trees of $F$, then let $f \leftarrow x$ and terminate the algorithm. Otherwise,
3. Find the preorder traversals of the three trees of $F$, and then let $f$ be the unique vertex that is both large and the rightmost element of the preorder traversal of some tree of $F$. with $k \leq 2$ missing edges
A new decoding algorithm

1. find the unique Hamiltonian path and label the vertices accordingly
2. find the fixed element \( f \)
3. find the set \( A \) of the child nodes of the root of the representative tree that are different from \( 2n+1 \)
4. calculate the key as follows

---

**Procedure 3: Constructing the set of large ascending vertices**

1. If \( F[X_c] \cup \{2n + 2\} \) is connected then \( A \leftarrow N_F(2n + 2) \) and terminate the algorithm. Otherwise,
2. If \( F[X_c] \cup \{2n + 2\} \) contains no isolated vertices then \( A \leftarrow N_F(2n + 2) \cup \{2n + 1\} \) and terminate the algorithm. Otherwise,
3. If \( F[X_c] \cup \{2n + 2\} \) contains two isolated vertices \( x, x' \) then \( A \leftarrow N_F(2n + 2) \cup \{x, x'\} \) and terminate the algorithm. Otherwise,
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 \( A \leftarrow N_F(2n + 2) \cup \{x, 2n + 1\} \)
     - else \( A \leftarrow N_F(2n + 2) \)
   - else \( A \leftarrow N_F(2n + 2) \cup \{x\} \)
A new decoding algorithm

1. find the unique Hamiltonian path and label the vertices accordingly
2. find the fixed element $f$
3. find the set $A$ of the child nodes of the root of the representative tree that are different from $2n+1$
4. calculate the key as follows

Procedure 3: Constructing the set of large ascending vertices

1. If $F[X_c] \cup \{2n + 2\}$ is connected then $A \leftarrow N_F(2n + 2)$ and terminate the algorithm. Otherwise,
2. If $F[X_c] \cup \{2n + 2\}$ contains no isolated vertices then $A \leftarrow N_F(2n + 2) \cup \{2n + 1\}$ and terminate the algorithm. Otherwise,
3. If $F[X_c] \cup \{2n + 2\}$ contains two isolated vertices $x, x'$ then $A \leftarrow N_F(2n + 2) \cup \{x, x'\}$ and terminate the algorithm. Otherwise,
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 $A \leftarrow N_F(2n + 2) \cup \{x, 2n + 1\}$
     else $A \leftarrow N_F(2n + 2)$
   else $A \leftarrow N_F(2n + 2) \cup \{x\}$

with $k \leq 2$ missing edges
A new decoding algorithm

1. find the unique Hamiltonian path and label the vertices accordingly
2. find the fixed element $f$
3. find the set $A$ of the child nodes of the root of the representative tree that are different from $2n+1$
4. calculate the key as follows with $k \leq 2$ missing edges
A new decoding algorithm

1. find the unique Hamiltonian path and label the vertices accordingly
2. find the fixed element $f$
3. find the set $A$ of the child nodes of the root of the representative tree that are different from $2n+1$
4. calculate the key as follows

**Experimental results**

<table>
<thead>
<tr>
<th>$n$ bits</th>
<th>former alg.</th>
<th>our alg.</th>
<th>our alg. (-1 edge)</th>
<th>our alg. (-2 edges)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>82.2 (4.4) μs</td>
<td>56.5 (3.2) μs</td>
<td>63.9 (6.7) μs</td>
<td>78.0 (16.4) μs</td>
</tr>
<tr>
<td>10</td>
<td>132.3 (9.3) μs</td>
<td>95.7 (5.8) μs</td>
<td>104.2 (9.4) μs</td>
<td>122.8 (24.8) μs</td>
</tr>
<tr>
<td>20</td>
<td>240.9 (11.8) μs</td>
<td>177.5 (9.7) μs</td>
<td>190.7 (17.4) μs</td>
<td>219.9 (44.9) μs</td>
</tr>
<tr>
<td>30</td>
<td>357.7 (14.4) μs</td>
<td>268.9 (13.2) μs</td>
<td>281.3 (18.2) μs</td>
<td>328.1 (66.0) μs</td>
</tr>
<tr>
<td>100</td>
<td>1406.7 (45.7) μs</td>
<td>1135.4 (39.5) μs</td>
<td>1151.2 (89.8) μs</td>
<td>1248.5 (260.4) μs</td>
</tr>
</tbody>
</table>

average time (standard deviation)
Our contribution

1. **formal definition of the class** of *canonical reducible permutation graphs* (precisely the graphs produced by Chroni and Nikolopoulos’s encoding algorithm)

2. **characterization** and **linear-time recognition algorithm** for such graphs

3. a **new linear-time decoding algorithm** (graph $\rightarrow$ integer key) simpler, marginally faster and able to retrieve the correct key even after the malicious removal of $k \leq 2$ edges

4. a **tight bound for the resilience of the codec** against edge removals
Resilience against edge modifications

\[ \omega = 2 \quad \text{(B = 10)} \]

\[ \omega = 3 \quad \text{(B = 11)} \]
Resilience against edge modifications

$\omega = 2 \quad (B = 10)$

$\omega = 3 \quad (B = 11)$
Resilience against edge modifications

\[ \omega = 2 \quad (B = 10) \]

\[ \omega = 3 \quad (B = 11) \]
Resilience against edge modifications

For \( n > 2 \) bits, it is possible to detect up to 5 edge insertions/deletions in polynomial time. This bound is tight.
Danke schön!

Vinícius Gusmão Pereira de Sá
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
vigusmao@dcc.ufrj.br
Towards a provably resilient scheme for graph-based watermarking

Lucila Maria Souza Bento
Davidson Boccardo
Raphael Carlos Santos Machado
Vinícius Gusmão Pereira de Sá
Jayme Luiz Szwarcfiter