Proteção de software por marcas d'água baseadas em grafos

Lucila Bento[1,3], Davidson Boccardo[3], Raphael Machado[3], Vinícius Pereira de Sá[1], Jayme Szwarcfiter[1,2,3]

[1]Instituto de Matemática – Universidade Federal do Rio de Janeiro

[2]COPPE – Universidade Federal do Rio de Janeiro

[3]INMETRO – Instituto Nacional de Metrologia, Qualidade e Tecnologia
Rio de Janeiro, Brazil

lucilabento@ppgi.ufrj.br, drboccardo@inmetro.gov.br,
rcmachado@inmetro.gov.br, vigusmao@dcc.ufrj.br, jayme@nce.ufrj.br

Abstract. *The insertion of watermarks into computer programs allows for the timely retrieval of authorship/ownership information, therefore discouraging software piracy. In this paper, we consider the graph-based watermarking scheme proposed by Chroni and Nikolopoulos (*An efficient graph codec system for software watermarking, COMPSAC'12*), and we formulate robust algorithms to restore a watermark from which $k > 0$ edges were maliciously removed. Moreover, we study the resilience of the considered scheme face to such kind of attack, and we present computational results indicating that the probability that a watermark cannot be restored after the removal of a fixed number k of edges tends to zero as the size of the watermark grows.*

Keywords: *software security, authenticity, watermark, graph*

Resumo. *A inserção de marcas d'água em programas de computador objetiva a posterior identificação de sua autoria ou propriedade, desencorajando a cópia ilegal dos mesmos. Neste artigo, consideramos o esquema de marcas d'água baseadas em grafos proposto por Chroni e Nikolopoulos (*An efficient graph codec system for software watermarking, COMPSAC'12*), e formulamos dois algoritmos robustos, com complexidades e aplicabilidades distintas, para a restauração de uma marca d'água da qual $k > 0$ arestas foram maliciosamente removidas. Ademais, estudamos a resiliência do esquema considerado diante desse tipo de ataque, e apresentamos resultados computacionais evidenciando que a probabilidade de uma marca d'água se tornar irrecuperável pela remoção de um número k fixo de arestas tende a zero à medida em que o tamanho da marca d'água aumenta.*

Palavras-chave: *segurança de software, autenticidade, marca d'água, grafo*

1. Introdução

A proteção à propriedade intelectual é objetivo incessantemente perseguido ao longo dos tempos. Em particular, a necessidade de se coibir a reprodução criminosa — pirataria — de software tem suscitado diversas frentes de pesquisa visando o desenvolvimento e o aprimoramento de medidas preventivas e fiscalizatórias.

Marcas d'água têm sido historicamente utilizadas[1] com a finalidade de introduzir informações de autoria ou propriedade em um objeto. A possibilidade de se aplicar marcas d'água a objetos digitais, especialmente software, foi há alguns anos descoberta. Por estar sujeita à manipulação por agentes maliciosos, são necessárias medidas para dificultar a identificação, remoção ou modificação da marca d'água. É importante também garantir que a aplicação da marca d'água não irá comprometer o funcionamento do software que se pretende proteger, e, preferencialmente, que seu impacto sobre o tamanho e o tempo de execução do programa seja insignificante.

Neste artigo, estamos interessados em técnicas que introduzem no grafo de fluxo de controle de um programa uma informação de identificação apropriadamente codificada: a marca d'água digital. Consideramos o esquema de marcas d'água baseadas em grafos proposto por Collberg, Carter, Kobourov e Thomborson [4] e posteriormente aprimorado por Chroni e Nikolopoulos [2, 3], que afirmaram sem provas que tal esquema seria resiliente a ataques que removessem uma aresta da marca d'água. Em trabalho recente, Bento *et al.* [1] demonstram formalmente a resiliência da marca d'água de Chroni e Nikolopoulos, apresentando um algoritmo que decodifica em tempo linear a informação contida na marca d'água, mesmo que até *duas* arestas tenham sido removidas, e provando que a decodificação nem sempre é possível diante de um número maior de remoções.

A caracterização completa, apresentada em [1], da classe de grafos que correspondem a marcas d'água codificadas pelo esquema de [2], permitiu a investigação de novos algoritmos para a recuperação de marcas d'água que sofreram a remoção maliciosa de k arestas, para qualquer inteiro positivo k. Tais algoritmos, apresentados na Seção 3, são robustos no sentido de que recuperam a marca d'água original sempre que possível, informando, de outra forma, que a marca d'água foi irremediavelmente corrompida (em razão de ter se tornado isomorfa a outra possível marca d'água com igual número de arestas removidas). Embora ambos os algoritmos sejam polinomiais no tamanho da marca d'água para qualquer valor fixo de k, suas complexidades de tempo diferem, beneficiando-se o algoritmo mais rápido do conhecimento dos rótulos dos vértices do grafo que compõe a marca d'água, ao passo que o mais lento prescinde dessa informação. Na Seção 4, apresentamos um algoritmo que permite recuperar os rótulos dos vértices quando $k \leq 3$. Finalmente, na Seção 5, comparamos experimentalmente os algoritmos decodificadores de [1, 2] e determinamos a probabilidade de que a remoção de k arestas corrompa irremediavelmente uma marca d'água. Os resultados obtidos indicam que, para um valor fixo de k, essa probabilidade tende a zero à medida em que o tamanho da marca d'água aumenta.

2. Conceitos preliminares e resultados anteriores

Marcas d'água baseadas em grafos. Em linhas gerais, um esquema de marcas d'água baseadas em grafos para proteção de software é composto por:

- um algoritmo de codificação, que converte um identificador (em geral, um número inteiro) em um grafo;
- um algoritmo de decodificação, que recupera o identificador a partir do grafo;
- uma *função embarcadora* (*embedder*), que recebe como entrada o programa onde será embarcada a marca d'água e o grafo correspondente à marca d'água desejada,

[1] A mais antiga marca d'água em papel conhecida data de 1292, e sua origem é a cidade italiana de Fabriano, fundamental na história da indústria do papel [7].

retornando um novo objeto (programa modificado) contendo a marca d'água;
- uma *função extratora*, que retorna a marca d'água embarcada no programa.

A todo programa de computador está associado um grafo direcionado, que determina suas possíveis sequências de passos. Tal grafo, denominado *grafo de fluxo*, pode ser recuperado a partir de uma análise estática do programa[2] (Figura 1).

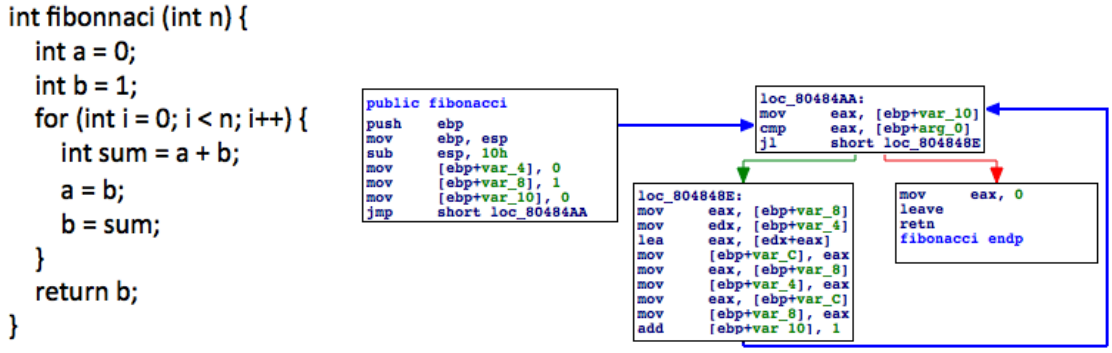

Figura 1. Programa em linguagem C++ e o grafo de fluxo correspondente

O trabalho pioneiro de Davidson e Myrhvold [6] propôs o uso do grafo de fluxo de um programa para codificar as informações de uma marca d'água, tendo inspirado diversos esquemas de marcas d'água baseadas em grafos desde então [5, 8, 9].

Ataques. Marcas d'água para software estão sujeitas aos seguintes tipos de ataque:

- *ataques de adição* — um agente malicioso inclui uma segunda marca d'água no programa, de forma a confundir a função extratora;
- *ataques de subtração* — o atacante descobre a localização da marca d'água e remove-a completamente;
- *ataques de distorção* — o atacante altera sintaticamente o programa por meio de transformações preservadoras de semântica, incapacitando a recuperação da marca d'água pela função extratora, que é em geral baseada na topologia do grafo de fluxo do programa, e portanto fortemente sensível à sintaxe.

Ataques de adição e de subtração podem ser contornados através de técnicas de criptografia e diversidade de software aplicadas sobre a informação a ser embarcada e sobre o programa onde a marca d'água será embarcada. Assim, no presente trabalho, estamos interessados em ataques de distorção, que, para todos os efeitos, significam a *remoção maliciosa de arestas* do grafo que compõe a marca d'água embarcada.

A marca d'água de Chroni e Nikolopoulos. Revisitamos agora, em breves linhas, a codificação proposta por Chroni e Nikolopoulos [3] para marcas d'água baseadas em grafos. Seja ω um inteiro positivo, e n o tamanho de sua representação binária B. Sejam n_0 e n_1 o número de 0's e 1's, respectivamente, em B. O binário estendido B^* é obtido pela concatenação de n dígitos 1, seguido do complemento-para-um de B (isto é, B com seus dígitos invertidos), e um dígito 0 adicional à direita. Denotamos por $n^* = 2n + 1$ o tamanho de B^*, e definimos $Z_0 = (z_i^0), i = 1, ..., n_1 + 1$, como a sequência ascendente

[2] Análise estática é aquela que não demanda a execução do programa.

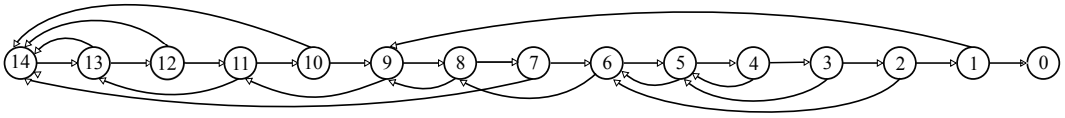

Figura 2. Marca d'água para a chave $\omega = 37$

de índices de 0's em B^*, e $Z_1 = (z_i^1), i = 1, ..., n + n_0$, como a sequência ascendente de índices de 1's em B^*.

A *permutação bitônica* $P_b = Z_0||Z_1^R = (b_i), i = 1, \ldots, n^*$, é obtida pela concatenação de Z_0 com os elementos de Z_1 em ordem inversa, de forma que a subsequência dos elementos de P_b é ascendente até uma posição k (com $k \leq n^*$) e descendente de k até n^*. Finalmente, a *permutação auto-inversível* P_a é obtida a partir de P_b da seguinte forma: para $i = 1, \ldots, n^*$, à posição de índice b_i, em P_a, é atribuído o elemento b_{n^*-i+1}, e à posição de índice b_{n^*-i+1}, em P_a, é atribuído o elemento b_i. Em outras palavras, os n pares não-ordenados de elementos distintos $(p, q) = (b_i, b_{n^*-i+1})$ de P_b, para $i = 1, \ldots, n$, são os chamados 2-*ciclos* de P_s, com p ocupando a posição q em P_a, e vice-versa. Como o índice $i = n + 1$ é a solução da equação $n^* - i + 1 = i$, o elemento central b_{n+1} de P_b aparecerá em P_a com índice igual ao próprio b_{n+1}, constituindo assim o único 1-*ciclo* de P_a, que é também chamado de *elemento fixo* de P_a.

A marca d'água obtida pelo esquema de Chroni e Nikolopoulos consiste em um grafo G tal que $V(G) = \{0, 1, ..., 2n+2\}$, e $E(G)$ possui $4n+3$ arestas, a saber: *arestas de caminho* $(u, u - 1)$ para todo $u = 1, \ldots, 2n + 2$, produzindo um caminho hamiltoniano que é único em G; e *arestas de árvore* de u para $q(u)$, $u = 1, \ldots, n^*$, onde $q(u)$ é definido como o vértice $v > u$ à esquerda de u e o mais próximo possível de u em P_a, se tal v existir, ou $2n + 2$, caso contrário. Um grafo assim obtido é um tipo especial de grafo de permutação redutível [4] chamado *grafo de permutação redutível canônico*, cuja caracterização formal foi apresentada em [1]. O vértice $2n + 2$ e o vértice 0 são, respectivamente, a única fonte (vértice com grau de entrada nulo) e o único sorvedouro (vértice com grau de saída nulo) de G.

Consideremos um exemplo. Para a chave $\omega = 37$, temos $B = 100101$, $n = 6$, $n_0 = 3$, $n_1 = 3$, $B^* = 1111110110100$, $n^* = 13$, $Z_0 = (7, 10, 12, 13)$, $Z_1 = (1, 2, 3, 4, 5, 6, 8, 9, 11)$, $P_b = (7, 10, 12, 13, 11, 9, 8, 6, 5, 4, 3, 2, 1)$ e $P_a = (7, 10, 12, 13, 11, 9, 1, 8, 6, 2, 5, 3, 4)$. A marca d'água associada a ω terá, ao longo de seu caminho hamiltoniano, as arestas $14 \rightarrow 13 \rightarrow \ldots \rightarrow 0$, e apresentará também as arestas de árvore $(1, 9), (2, 6), (3, 5), (4, 5), (5, 6), (6, 8), (7, 14), (8, 9), (9, 11), (10, 14), (11, 13), (12, 14)$ e $(13, 14)$, como ilustrado pela Figura 2.

3. Algoritmos polinomiais robustos para a recuperação de marcas d'água

A caracterização de Bento *et al.* para grafos de permutação redutíveis canônicos permite identificar se um grafo corresponde a uma marca d'água válida em tempo linear [1], abrindo caminho para a construção de algoritmos robustos para a recuperação de marcas d'água que sofreram a remoção de k arestas, para qualquer valor de k. Os algoritmos que propomos nesta seção fazem exatamente isso, e são robustos no sentido de que recuperam *sempre que possível* a marca d'água danificada, ou, de outra forma, informam que existe mais de uma *marca d'água candidata*, isto é, que mais de uma marca d'água

poderia ter originado a marca d'água danificada (pela perda de k de suas arestas). Os algoritmos a seguir diferem com relação aos seus dados de entrada: enquanto o primeiro deles (apresentado aqui de duas formas distintas, apenas por clareza de exposição) recebe como entrada apenas a estrutura do grafo associado à marca d'água danificada, o segundo necessita que os vértices desse grafo estejam "rotulados" de acordo com o caminho hamiltoniano da marca d'água original, sendo capaz de tirar vantagem dessa informação adicional para reduzir sua complexidade de tempo em relação ao primeiro algoritmo.

Um primeiro algoritmo. Seja G uma marca d'água e G' o grafo obtido de G após a remoção de k arestas. O algoritmo mais intuitivo — ou ingênuo — para a recuperação de G consiste em simplesmente adicionarmos a G', um por vez, cada um dos possíveis conjuntos de k não-arestas de G', verificando, para cada escolha, se uma marca d'água válida foi assim obtida. Observe que reconhecer se um grafo pertence à classe de grafos associados a marcas d'água de Chroni e Nikolopoulos (isto é, à classe dos grafos de permutação redutíveis canônicos [1]) é tarefa que pode ser realizada em tempo linear por algoritmo que decorre diretamente da caracterização de tais grafos apresentadas em [1]. Como $|V(G')| = 2n + 3$ e $|E(G')| = 4n + 3 - k$, a quantidade de subconjuntos de k não-arestas de G' é igual a $\binom{(2n+3)(2n+2)/2-(4n+3-k)}{k} = O(n^{2k})$. Portanto, a complexidade total do algoritmo é $O(n^{2k}) \cdot O(n) = O(n^{2k+1})$.

Aprimorando o algoritmo. O algoritmo descrito acima é um tanto grosseiro ao considerar que qualquer das não-arestas de uma marca d'água danificada G' poderia ser uma aresta da marca d'água original G. Na verdade, devido à estrutura bem definida — e bastante restrita — de G, relativamente poucas dentre essas não-arestas de G' poderiam realmente pertencer a G. Mais precisamente, sabemos que quase todos os vértices de G possuem grau de saída exatamente igual a 2, a menos do vértice-fonte e do vértice-sorvedouro, cujos graus de saída devem ser 1 e 0, respectivamente. Dessa forma, cada vértice $v \in V(G')$ contribui com $0 \leq 2 - |N^+_{G'}(v)| \leq 2$ elementos para o multiconjunto M^* contendo todos os candidatos a origens de arestas removidas de G. É fácil ver que $|M^*| = 2 \cdot |V(G')| - |E(G')| = 2 \cdot (2n + 3) - (4n + 3 - k) = k + 3$, de forma que as k origens podem ser escolhidas de $\binom{|M^*|}{k} = O(k^3)$ maneiras. Uma vez escolhido um subconjunto $M \subseteq M^*$ de k origens, passamos aos vértices de destino. Cada origem tem $O(n)$ destinos possíveis, num total de $O(n^k)$ possibilidades. Considerando a execução do algoritmo linear de reconhecimento para cada uma dessas possíveis escolhas, chegamos à complexidade $O(k^3) \cdot O(n^k) \cdot O(n) = O(k^3 \cdot n^{k+1})$ para todo o algoritmo. Se exatamente uma marca d'água foi reconhecida nesse processo, a recuperação foi bem-sucedida. Caso contrário, terá ficado provada a impossibilidade da recuperação pela exibição de um conjunto $\mathcal{L}$ contendo todas as marcas d'água que poderiam ter originado G'.

Segundo algoritmo: fazendo uso da informação dos rótulos. O segundo algoritmo que propomos para a recuperação de marcas d'água parte do princípio de que os rótulos de todos os vértices da marca d'água são conhecidos — ou podem ser determinados — *a priori*. Note que, por construção, conhecer os rótulos dos vértices equivale a conhecer todas as arestas do caminho hamiltoniano de G. Para efeito de análise, portanto, suporemos que as k arestas removidas foram arestas de árvore de G, isto é, arestas que não são da forma $(j, j - 1)$ na marca d'água original G.

Procedimento 1: Reconstrução da marca d'água

Entrada: grafo $G'(V, E')$
Saída: lista $\mathcal{L}$ de marcas d'água candidatas

1. $\mathcal{L} \leftarrow \emptyset$
2. determine o número de arestas removidas $k \leftarrow 2|V| + 1 - |E|$
3. multiconjunto $\mathcal{S} \leftarrow \{v \mid v \in V, |N^+_{G'}(v)| = 1\} \cup \{v, v \mid v \in V, |N^+_{G'}(v)| = 0\}$
4. seja $\mathcal{E}' \leftarrow \mathcal{S} \times V$
4. **para cada** conjunto $R = \{e_1, \ldots, e_k\} \subset \mathcal{E}$
4.1. **se** $G = (V, E' \cup R)$ é uma marca d'água **então** adicione G a $\mathcal{L}$
5. **retorne** $\mathcal{L}$ // se $|\mathcal{L}| = 1$, a marca d'água original foi recuperada

Sabemos que o vértice de rótulo $2n + 2$ é a raiz da *árvore representativa* T formada por todas as arestas de árvore de G [1]. Uma vez que, por hipótese, k arestas foram removidas de T, o subgrafo F formado pelos vértices de G' (com exceção do vértice-sorvedouro, de rótulo 0) e por todas as arestas de árvore de G' apresentará exatamente $k + 1$ componentes conexos. É também sabido, pela propriedade *max-heap* das árvores representativas (pela qual se u é filho de v em T então $u < v$) [1], que a raiz de cada sub-árvore T' de T é precisamente o vértice de maior rótulo em T'. Ficam, dessa forma, determinadas as origens das arestas removidas de G': os vértices v de maior rótulo em cada componente conexo de F, excetuando-se o componente que contém $2n + 2$. Definidas as origens, basta escolher, para cada origem v, uma aresta (v, w), com $w > v$, e acrescentá-la a G', verificando, no final, se o grafo assim obtido constitui uma marca d'água válida.

Procedimento 2: Reconstrução da marca d'água usando os rótulos

Entrada: grafo $G'(V, E')$
Saída: lista $\mathcal{L}$ de marcas d'água candidatas

1. $\mathcal{L} \leftarrow \emptyset$
2. $E'' \leftarrow E' \setminus \{(2n + 2, 2n + 1), (2n + 1, 2n), \ldots, (1, 0)\}$ // E'' recebe as arestas de árvore
3. $F \leftarrow (V, E'')$
4. $M \leftarrow \emptyset$ // origens das arestas removidas
5. **para cada** componente conexo $C \not\supset \{2n + 2\}$ de F
5.1. adicione a M o vértice v de maior rótulo em C
6. **para cada** $R = \{e_1, \ldots, e_k\}$, com $e_i = (v_i, w) \notin E'$, $v_i \in M$, $v_i < w \in V$ $(i = 1, \ldots, k)$
6.1. **se** $G(V, E' \cup R)$ é uma marca d'água **então** adicione G a $\mathcal{L}$
7. **retorne** $\mathcal{L}$ // se $|\mathcal{L}| = 1$, a marca d'água original foi recuperada

Para a análise de complexidade, observe que cada conjunto R possui k elementos e corresponde a uma atribuição de destinos às origens das arestas removidas. Como cada aresta tem $O(n)$ destinos possíveis, o número de tais atribuições é $O(n^k)$, cada uma das quais correspondendo a um grafo que será testado, em tempo $O(n)$, quanto a ser ou não uma marca d'água válida, perfazendo a complexidade de $O(n^{k+1})$ para todo o algoritmo.

4. Reconstrução do caminho hamiltoniano

Além do ganho em desempenho proporcionado pelo conhecimento dos rótulos dos vértices (eliminando um fator de $k!$ da complexidade da recuperação robusta de marcas d'água com k arestas a menos), a restauração do caminho hamiltoniano de uma marca d'água danificada abre portas para o desenvolvimento de algoritmos ainda mais rápidos

para valores fixos de k. Em [1], por exemplo, os rótulos obtidos pela reconstituição do caminho hamiltoniano tem papel central no algoritmo de tempo linear que foi proposto para o caso $k \leq 2$. Formulamos a seguir um algoritmo, descrito em pseudo-código como Procedimento 3, para a recuperação do caminho hamiltoniano H de uma marca d'água G, após $k = 3$ arestas de G terem sido removidas. Como nem todas as arestas removidas são necessariamente arestas do caminho hamiltoniano, o algoritmo classifica as arestas removidas como arestas de caminho, caso pertençam a H, ou arestas de árvore, caso pertençam à árvore representativa $T = G \setminus H$.

O algoritmo tem por entrada uma marca d'água danificada $G' = G \setminus \{e_1, e_2, e_3\}$. Se $v \in V(G')$, denotamos por $H(v)$ o subcaminho de H que termina em v e que é iniciado o mais longe possível de v em G'. Denotamos por $N^+_{G'}(H(v))$ e $N^-_{G'}(H(v))$ a vizinhança de saída e entrada de $H(v)$ em G', respectivamente. E, finalmente, denotamos por $primeiro(H(v))$ o vértice inicial do subcaminho terminado em v.

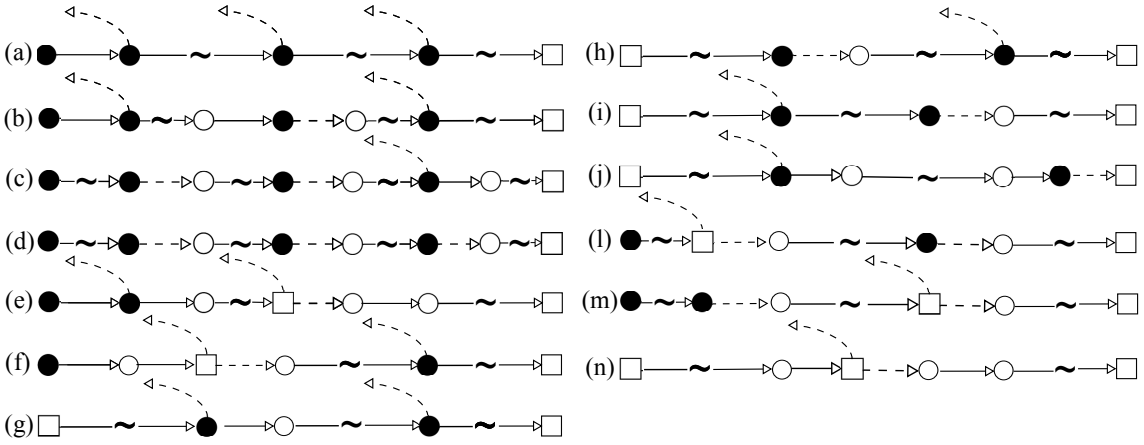

Figura 3. Cenários possíveis após a remoção de três arestas. Linhas tracejadas indicam arestas removidas. Linhas interrompidas (com um til) indicam caminhos de tamanho arbitrário. Quadrados, círculos pretos e círculos brancos representam vértices com grau de saída 0, 1 e 2, respectivamente.

Consideremos, em linhas gerais, as possíveis alternativas computadas pelo Procedimento 3. Observe que cada linha com uma instrução "**retorne** H" é acompanhada por um comentário indicando a situação correspondente na Figura 3. Se o algoritmo termina descrevendo a situação da Figura 3(a), então H foi preservado pela atuação do atacante, e não há nada a fazer. Por outro lado, terminando nas situações das Figuras 3(b–n), uma ou mais arestas do caminho hamiltoniano de G foram de fato removidas pelo ataque e não estão presentes em G'. O restabelecimento do caminho completo, em cada um dos casos, envolve a concatenação, na ordem correta, dos caminhos maximais que podem ser obtidos, iterativamente, cada qual terminando em um vértice v que apresenta grau de saída nulo no subgrafo de G' induzido pelos vértices que ainda não apareceram em nenhum subcaminho determinado até então. Como é possível observar, há casos em que não é possível recuperar o caminho hamiltoniano com unicidade, e o algoritmo retorna falha.[3]

Uma vez que $1 \leq |V_0| \leq 3$, $1 \leq |V_1| \leq 4$, e que cada subcaminho $H(v_i)$ pode ser computado facilmente em tempo $|H(v_i)|$, todo o algoritmo roda em tempo linear $O(n)$.

[3]A análise formal de corretude do Procedimento 3 é longa e será apresentada oportunamente numa versão estendida deste texto.

Procedimento 3: Reconstrução do caminho hamiltoniano (caso $k = 3$)

Entrada: grafo $G'(V, E')$
Saída: o caminho hamiltoniano único da marca d'água G que originou G'

1. $V_0 \leftarrow \{v \in V(G') \text{ tal que } |N^+_{G'}| = 0\}$
2. $V_1 \leftarrow \{v \in V(G') \text{ tal que } |N^+_{G'}| = 1\}$
3. **se** $|V_0| = 1$ **então**
3.1. seja $V_0 = \{v_0\}$
3.2. **se** $|H(v_0)| = 2n + 3$ **então**
3.2.1. $H \leftarrow H(v_0)$, **retorne** H // Figura 3(a)
3.3. **senão se** $\exists\, v_1 \in V_1$ tal que $|H(v_1)| + |H(v_0)| = 2n + 3$ **então**
3.3.1. $H \leftarrow H(v_1) \,||\, H(v_0)$, **retorne** H // Figura 3(b)
3.4. **senão se** $\exists\, v_1, v'_1 \in V_1$ tal que $|H(v'_1)| + |H(v_1)| + |H(v_0)| = 2n + 3$ **então**
3.4.1. $H \leftarrow H(v'_1) \,||\, H(v_1) \,||\, H(v_0)$, **retorne** H // Figura 3(c)
3.5. **senão**
3.5.1. seja $V_1 = \{v_1, v'_1, v''_1\}$
3.5.2. **se** $|H(v''_1)| + |H(v'_1)| + |H(v_1)| + |H(v_0)| = 2n + 3$ **então**
3.5.2.1. **se** $N^+_{G'}(H(v_1)) \cap H(v''_1) \neq \emptyset$ **então**
3.5.2.1.1. $H \leftarrow H(v'_1) \,||\, H(v''_1) \,||\, H(v'_1) \,||\, H(v_0)$, **retorne** H // Figura 3(d)
3.5.2.2. **senão se** $N^+_{G'}(H(v''_1)) \cap H(v_1) \neq \emptyset$ **então**
3.5.2.2.1. $H \leftarrow H(v'_1) \,||\, H(v_) \,||\, H(v''_1) \,||\, H(v_0)$, **retorne** H // Figura 3(d)
3.5.2.3. **senão se** $|H(v'_1)| = 3$ **e** $N^+_{G'}(H(v''_1)) \cap primeiro(H(v'_1)) \neq \emptyset$ **e** $N^+_{G'}(H(v_1)) \cap v'_1 \neq \emptyset$ **então**
3.5.2.3.1. $H \leftarrow H(v'_1) \,||\, H(v''_1) \,||\, H(v_1) \,||\, H(v_0)$, **retorne** H // Figura 3(d)
3.5.2.4. **senão se** $|H(v'_1)| = 3$ **e** $N^+_{G'}(H(v_1)) \cap primeiro(H(v'_1)) \neq \emptyset$ **e** $N^+_{G'}(H(v''_1)) \cap v'_1 \neq \emptyset$ **então**
3.5.2.4.1. $H \leftarrow H(v'_1) \,||\, H(v_1) \,||\, H(v''_1) \,||\, H(v_0)$, **retorne** H // Figura 3(d)
3.5.2.5. **senão** impossível recuperar
4. **senão se** $|V_0| = 2$ **então**
4.1. seja $V_0 = \{v_0, v'_0\}$
4.2. **se** $|H(v'_0)| + |H(v_0)| = 2n + 3$ **então**
4.2.1. seja v_0 tal que $N^+_{G'}(H(v_0)) \cap H(v'_0) \neq \emptyset$
4.2.2. $H \leftarrow H(v'_0) \,||\, H(v_0)$, **retorne** H // Figura 3(e, f, g)
4.3. **senão se** $\exists\, v_1 \in V_1$ tal que $|H(v'_0)| + |H(v_1)| + |H(v_0)| = 2n + 3$
4.3.1. identifique v'_0; $H \leftarrow H(v'_0) \,||\, H(v_1) \,||\, H(v_0)$, **retorne** H // Figura 3(h, i, j)
4.4. **senão se** $N^+_{G'}(H(v_1)) \cap H(v'_0) \neq \emptyset$ **então**
4.4.1. $H \leftarrow H(v'_0) \,||\, H(v_1) \,||\, H(v_0)$, **retorne** H // Figura 3(l)
4.5. **senão se** $N^+_{G'}(H(v_0)) \cap H(v'_0) \neq \emptyset$ **então**
4.5.1. $H \leftarrow H(v_1) \,||\, H(v'_0) \,||\, H(v_0)$, **retorne** H // Figura 3(m)
4.6. **senão se** $N^+_{G'}(H(v'_0)) \cap H(v_0) \neq \emptyset$ **então**
4.6.1. $H \leftarrow H(v_1) \,||\, H(v_0) \,||\, H(v'_0)$, **retorne** H // Figura 3(m)
4.7. **senão se**$|H(v_1)| = 3$ **e** $N^+_{G'}(H(v'_0)) \cap primeiro(H(v_1)) \neq \emptyset$ **e** $N^+_{G'}(H(v_0)) \cap v_1 \neq \emptyset$ **então**
4.7.1. $H \leftarrow H(v_1) \,||\, H(v'_0) \,||\, H(v_0)$, **retorne** H // Figura 3(m)
4.8. **senão se**$|H(v_1)| = 3$ **e** $N^+_{G'}(H(v_0)) \cap primeiro(H(v_1)) \neq \emptyset$ **e** $N^+_{G'}(H(v'_0)) \cap v_1 \neq \emptyset$ **então**
4.8.1. $H \leftarrow H(v_1) \,||\, H(v_0) \,||\, H(v'_0)$, **retorne** H // Figura 3(m)
4.9. **senão** impossível recuperar
5. **senão**
5.1. $sejaV_0 = \{v_0, v'_0, v''_0\}$
5.2. $H \leftarrow H(v'_0) \,||\, H(v''_0) \,||\, H(v_0)$, **retorne** H // Figura 3(n)

5. Resultados computacionais

O objetivo desta seção é o de enriquecer o entendimento do esquema de marca d'água proposto por Chroni e Nikolopoulos [3], observando o comportamento dos algoritmos de

Tabela 1. Tempos médios (e desvios-padrão) de decodificação.

n (bits)	alg. de [3] (sem remoções)	alg. de [1] (sem remoções)	alg. de [1] (-1 aresta)	alg. de [1] (-2 arestas)
5	82.2 (4.4) μs	56.5 (3.2) μs	63.9 (6.7) μs	78.0 (16.4) μs
10	132.3 (9.3) μs	95.7 (5.8) μs	104.2 (9.4) μs	122.8 (24.8) μs
20	240.9 (11.8) μs	177.5 (9.7) μs	190.7 (17.4) μs	219.9 (44.9) μs
30	357.7 (14.4) μs	268.9 (13.2) μs	281.3 (18.2) μs	328.1 (66.0) μs
100	1406.7 (45.7) μs	1135.4 (39.5) μs	1151.2 (89.8) μs	1248.5 (260.4) μs

decodificação propostos em [3] e [1], e determinando a probabilidade de que um ataque a k arestas seja bem sucedido. Primeiramente, foram implementados os algoritmos de codificação e de decodificação exatamente como descritos em [3], assim como o algoritmo de decodificação apresentado em [1]. Foram, a seguir, geradas as marcas d'água referentes ao conjunto W formado por:

- todas as chaves nos intervalos $[2^4, 2^5 - 1]$ e $[2^9, 2^{10} - 1]$,
- 10 mil chaves escolhidas aleatória e uniformemente no intervalo $[2^{19}, 2^{20} - 1]$,
- 10 mil chaves escolhidas aleatória e uniformemente no intervalo $[2^{29}, 2^{30} - 1]$, e
- 10 mil chaves escolhidas aleatória e uniformemente no intervalo $[2^{99}, 2^{100} - 1]$.

Passemos agora aos testes realizados.[4]

5.1. Comparativo dos algoritmos de decodificação

O primeiro teste considerou a decodificação de marcas d'água íntegras. Com efeito, todas as marcas d'água de W foram utilizadas como entrada para os algoritmos de decodificação estudados. Os dois algoritmos exibiram comportamento nitidamente linear, rodando em tempo $O(n)$ para marcas d'água de tamanho $\Theta(n)$, tal como previsto analiticamente. O algoritmo proposto por Bento *et al.* [1], embora dotado da propriedade de se recuperar de remoções de até 2 arestas, rodou marginalmente mais rápido (em torno de 20%) do que o de Chroni e Nikolopoulos [3], que não possui a mesma propriedade. Os resultados podem ser vistos nas duas primeiras colunas da Tabela 1.

O segundo teste considerou a decodificação, pelo algoritmo de [1], de marcas d'água das quais $1 \leq k \leq 2$ arestas foram removidas. A partir de um número suficientemente grande de marcas d'água tomadas aleatoriamente de cada um dos subconjuntos de W descritos anteriormente, totalizando um mínimo de 20 mil instâncias por valor de n (ou exaurindo todos os casos possíveis, nos casos $n = 5$ e $n = 10$), foram removidos, um por vez, todos os subconjuntos de $k \in \{1, 2\}$ arestas de cada uma daquelas marcas d'água. Procedemos, a seguir, às decodificações de cada uma das instâncias assim obtidas, e o algoritmo foi capaz de recuperar a marca d'água original em todos os casos testados. A terceira e a quarta coluna da Tabela 1 apresentam os tempos de decodificação médios que foram obtidos, permitindo concluir que foi irrisório o aumento no tempo de execução decorrente de toda a etapa adicional — para a detecção de ataques e reconstituição das marcas d'água atacadas — executada antes da decodificação propriamente dita.

[4] Acesse em `https://www.dropbox.com/s/5tx8h3e72016bo6/watermarking.py` o código-fonte, escrito em linguagem Python. Todos os resultados foram obtidos em uma máquina iMac 2.9 GHz Intel Core i5 com 8 GB de RAM.

5.2. A resiliência da marca d'água de Chroni e Nikolopoulos

O algoritmo de decodificação de Bento *et al.* [1] permite a recuperação de marcas d'água (para todas as chaves com $n > 2$ bits) que sofreram a remoção de até duas arestas. Já era também conhecido o fato de não ser sempre possível a recuperação de uma marca d'água que sofreu a remoção de três ou mais arestas, uma vez que existem diversos pares de marcas d'água que se distinguem por não mais do que três arestas. Não obstante, para melhor compreendermos a resiliência do esquema considerado a ataques de distorção, importa conhecer a probabilidade de que seja bem-sucedido — no sentido de impedir a recuperação da marca d'água — um ataque pelo qual $k \geq 3$ arestas são removidas.

Seja $s(n,k)$ o número de marcas d'água G que codificam chaves de tamanho n e para as quais existe alguma marca d'água G' de mesmo tamanho tal que $E_T(G') \setminus E_T(G) = E_T(G) \setminus E_T(G') \leq k$, onde o subscrito T indica que estamos considerando os conjuntos de arestas de árvore apenas (marcas d'água G, G' que satisfazem essa desigualdade são ditas k-*sinônimas*). Seja $r(n,k) = s(n,k)/t(n)$ a razão entre $s(n,k)$ e o total $t(n) = 2^{n-1}$ de marcas d'água distintas para chaves de tamanho n. Finalmente, seja $p(n,k)$ a probabilidade de que a remoção de até k arestas de árvore de uma marca d'água referente a chave de n bits a torne irrecuperável (por deixá-la isomorfa a alguma outra marca com até k arestas de árvore removidas). Considere, para o cálculo de $p(n,k)$, o seguinte experimento:

Experimento A: Escolha, aleatória e uniformemente, uma marca d'água G_j do conjunto $M_n = \{G_1, G_2, \ldots, G_{t(n)}\}$ de todas as marcas d'água associadas a chaves de tamanho n. Escolha, a seguir, também de forma aleatória e uniforme, um subconjunto K contendo k arestas do conjunto de $2n+1$ arestas de árvore de G_j.

Queremos obter a probabilidade $p(n,k)$ de que o grafo $G_j \setminus K$ seja isomorfo de $G_{j'} \setminus K'$, para algum grafo $G_{j'} \neq G_j$ pertencente a M_n e algum subconjunto K' de k arestas de árvore de $G_{j'}$. Apesar de ser natural a formulação do experimento acima, os cálculos e a obtenção dos dados experimentais serão facilitados se considerarmos um experimento análogo para a escolha de G_j e K. Seja $\mathcal{K}_i$ a coleção de todos os subconjuntos de k arestas de árvore de G_i, para i de 1 a $t(n)$, e seja $\mathcal{K}$ a união de todos os $\mathcal{K}_i$.

Experimento B: Escolha, aleatória e uniformemente, um conjunto $K \in \mathcal{K}$.

Observe que cada elemento de $\mathcal{K}$ é um subconjunto de k arestas *de um determinado grafo* $G_i \in M_n$. Em outras palavras, uma vez que os conjuntos de vértices dos grafos considerados são mutuamente disjuntos, os $\mathcal{K}_i$ são todos também disjuntos dois a dois. Dessa forma, a escolha de K determina unicamente a marca d'água $G_j \in M_n$ que contém K, e cada marca d'água G_j tem probabilidade idêntica de ser dessa forma escolhida. Em outras palavras, os experimentos A e B são equivalentes.

A probabilidade $p(n,k)$ desejada é, portanto, simplesmente a probabilidade de que o subconjunto K escolhido pelo experimento B seja um subconjunto de k arestas que, quando removido da marca d'água G_j que o contém, torna G_j isomorfa a alguma outra marca d'água de mesmo tamanho e também com k arestas a menos. Definimos $q(n,k)$ como a quantidade de tais subconjuntos. Como K é escolhido uniformemente de $\mathcal{K}$, a probabilidade desejada fica determinada pela razão entre $q(n,k)$ e a quantidade de elementos de $\mathcal{K}$.

O Procedimento 4, apresentado a seguir, obtém o numerador $q(n,k)$ da razão definida acima. Na linha 2.2, o número binomial representa a quantidade de maneiras de se escolher um conjunto de k arestas de forma a tornar a marca d'água G_j isomorfa à marca d'água $G_{j'}$, que possui exatamente $d \leq k$ arestas distintas das arestas de G_j (fixamos as d arestas, escolhemos livremente as demais); o fator 2 se deve ao fato de serem duas marcas d'águas sinônimas, e portanto duas contribuições distintas ao somatório $q(n,k)$.

Procedimento 4: Determinação de $q(n,k)$

1. $q \leftarrow 0$
2. **para todo** par de marcas d'água $G_j, G_{j'} \in M_n$, com $j < j'$
2.1. $d \leftarrow |E_t(G_j) \setminus E_t(G_{j'})|$ // $E_t(G)$ é o conjunto de arestas de árvore de G
2.2. **se** $d \leq k$ **então** $q \leftarrow q + 2 \cdot \binom{2n+1-d}{k-d}$
3. **retorne** q

Para o denominador da razão em que estamos interessados, podemos escrever $|\mathcal{K}| = t(n) \cdot \binom{2n+1}{k} = 2^{n-1} \cdot \binom{2n+1}{k}$. A probabilidade desejada fica, portanto, determinada por $p(n,k) = \frac{q(n,k)}{|\mathcal{K}|}$, onde numerador e denominador já são conhecidos.

Para o terceiro e último teste, portanto, foram geradas as marcas d'água correspondentes a todas as chaves de n bits e determinados os valores $r(n,k)$ e $p(n,k)$ para todos os pares $(n,k) \in [3,15] \times [2,4]$. Observe que, apesar de terem sido os valores $r(n,k)$ e $p(n,k)$ obtidos considerando-se apenas a remoção de arestas de árvore, esses valores são claramente limites inferiores e superiores, respectivamente, para o caso geral em que k arestas — dentre arestas de árvore e arestas de caminho — foram removidas. A Tabela 2 apresenta os resultados encontrados. Embora a razão $r(n,k)$ aumente com n (o que é natural, dado o crescimento exponencial da quantidade de marcas d'água para chaves de tamanho n, em contraste com o crescimento apenas quadrático do número de possíveis arestas de árvore), a probabilidade de que um ataque a k arestas seja bem sucedido decresce com n, o que é extremamente favorável à confiabilidade do esquema considerado.

6. Considerações Finais

Uma questão relevante é o quão razoável é assumir que estão disponíveis os rótulos dos vértices de uma marca d'água. Há várias situações em que se pode assumir isto. A primeira situação acontece quando, embora algumas arestas tenham sido removidas, o caminho hamiltoniano permaneceu intacto — fato que é de fácil verificação, bastando executar uma busca no grafo a partir do único de seus vértices que terá grau de saída nulo. Uma segunda situação advém do fato de que os rótulos podem estar explicitamente indicados nos blocos de código do programa associados aos vértices do grafo. Ou seja, os rótulos não mais seriam determinados através do caminho hamiltoniano único do grafo, mas a partir de uma análise dos blocos de código do programa, os quais trariam codificados (preferencialmente, de maneira furtiva) tais rótulos. Finalmente, a terceira situação corresponde aos casos em que é possível recuperar o caminho hamiltoniano danificado — e, consequentemente, os rótulos dos vértices. Reveste-se, portanto, de grande importância a capacidade de se recuperar o caminho hamiltoniano de uma marca d'água danificada, sendo esta uma direção promissora a ser trilhada em pesquisas futuras.

Tabela 2. Frequência relativa $r(n,k)$ de marcas d'água referentes a chaves de tamanho n e k-sinônimas de alguma outra marca d'água, e probabilidade $p(n,k)$ de uma marca d'água se tornar irrecuperável após a remoção de k arestas.

n (bits)	$r(n,3)$	$r(n,4)$	$r(n,5)$	$p(n,3)$	$p(n,4)$	$p(n,5)$
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%

Referências

[1] L. Bento, D. Boccardo, R. Machado, V. Pereira de Sá, J. Szwarcfiter (2013). Towards a provably robust scheme for graph-based software watermarking. `http://arxiv.org/abs/1302.7262`.

[2] M. Chroni, S. D. Nikolopoulos (2011). Efficient encoding of watermark numbers as reducible permutation graphs. `http://arxiv.org/abs/1110.1194`.

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

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

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

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

[7] M. Kutter, F. Hartung (2000). Introduction to watermarking techniques. In S. Katzenbeisser and F. Petitcolas, editors, Information Hiding: Techniques for Steganography and Digital Watermarking, 97–120, Artech House.

[8] R. Venkatesan, V. Vazirani, S. Sinha (2001). A graph theoretic approach to software watermarking. Proc. 4th International Information Hiding Workshop, 157–168.

[9] J. Zhu, Y. Liu, K. Yin (2009). A novel dynamic graph software watermark scheme. Proc. 1st Int'l Workshop on Education Technology and Computer Science 3, 775–780.