Resilient Information Hiding
for Abstract Semi-Structures [*]

Radu Sion, Mikhail Atallah, Sunil Prabhakar

Computer Sciences Department and
The Center for Education and Research in Information Assurance,
Purdue University, West Lafayette, IN, 47907, USA,
`[sion, mja, sunil]@cs.purdue.edu`

Abstract. Most work on watermarking has resulted in techniques for different types of data: image, audio, video, text/language, software, etc. In this paper we discuss the watermarking of abstract structured aggregates of multiple types of content, such as multi-type/media documents. These *semi-structures* can be usually represented as graphs and are characterized by value lying both in the structure *and* in the individual nodes. Example instances include XML documents, complex web content, workflow and planning descriptions, etc. We propose a scheme for watermarking abstract semi-structures and discuss its resilience with respect to attacks. While content specific watermarking deals with the issue of protecting the value in the structure's nodes, protecting the value pertaining to the structure itself is a new, distinct challenge. Nodes in semi-structures are value-carrying, thus a watermarking algorithm could make use of their encoding capacity by using traditional watermarking. For example if a node contains an image then image watermarking algorithms can be deployed for that node to encode parts of the global watermark. But, given the intrinsic value attached to it, the graph that "glues" these nodes together is in itself a central element of the watermarking process we propose. We show how our approach makes use of these two value facets, structural and node-content.

1 Introduction

Digital Watermarking, in the traditional sense [5] [8] can be summarized as a steganographic technique embedding un-detectable (un-perceivable) hidden information into media objects (i.e. images, audio, video, text) with the main purpose of protecting the data from unauthorized duplication and distribution by enabling provable ownership. More recent results take on various other data domains such as natural language processing [1], software [4] [10] and relational data [13], [11]. Here we introduce an algorithm for watermarking abstract structured aggregates of multiple types of content that can be usually represented as graphs and are characterized by value lying

---

both in the structure *and* in the individual nodes (e.g. XML documents, complex Web Content, workflow and planning descriptions).

Most media watermarking techniques make use of the inherent large noise-bandwidth associated with Works that are to be "consumed" by the human sensory system with its limitations. In the case of abstract semi-structures, watermarking bandwidth appears to be available from capacities associated to properties of both the graph and the composing nodes. This introduces a whole new set of challenges and associated trade-offs. The trade-off between a required level of mark resilience and the ability to maintain guaranteed error bounds, structural equivalence and higher level semantics needs to be explored. Our solution is based on a canonical labeling algorithm that self-adjusts to the specifics of the content. Labeling is tolerant to a significant number of graph attacks ("surgeries") and relies on a complex "training" phase at watermarking time in which it reaches an optimal stability point with respect to these attacks. We perform attack experiments on the introduced algorithms under different conditions.

The paper is structured as follows. Section 2 is dedicated to a more in depth presentation of generic issues associated with watermarking in the framework of semi-structures. We analyze associated challenges and discuss attacks. Section 3 introduces important building blocks and concepts for the presented semi-structure watermarking algorithm, such as *tolerant canonical labeling* and *tolerant content summaries*. It presents and analyzes the main algorithm. Section 4.2 discusses experimental results. The **wmx.*** package is introduced. Section 5 concludes.

2 Challenges

2.1 The Model

One fundamental difference between watermarking and generic data hiding resides in the main applicability and descriptions of the two domains. Data hiding in general and covert communication in particular, aims at enabling Alice and Bob to exchange messages in a manner as stealthy as possible, through a medium controlled by evil Mallory. On the other hand, digital watermarking (especially for rights assessment) is deployed by Alice to prove rights over a piece of data, to Jared the Judge, usually in the case when Mallory benefits from using/selling that very same piece of data or maliciously modified versions of it.

In digital watermarking, the actual value to be protected lies in the Works themselves whereas information hiding usually makes use of them as simple value "transporters". Rights assessment can be achieved by demonstrating that a particular Work exhibits a rare property (read "hidden message" or "watermark"), usually known only to Alice (with the aid of a "secret" - read "watermarking key"). For court convince-ability purposes this property needs to be so rare that if one considers any other random Work "similar enough" to the one in question, this property is "very improbable" to apply (i.e. bound on false-positives). There is a threshold determining Jared's convince-ability related to the "very improbable" assessment. This defines a main difference from steganography: from Jared's perspective, specifics of the property (e.g. watermark message) are irrelevant as long as Alice can prove "convincingly" it is she who embedded/induced it to the original (non-watermarked) Work.

It is to be stressed here this particularity of watermarking for rights assessment. In watermarking the emphasis is on "detection" rather than "extraction". Extraction of a watermark (or bits of it) is usually a part of the detection process but just complements the process up to the extent of increasing the ability to convince in court. If recovering the watermark data in itself becomes more important than detecting the actual existence of it (i.e. 'yes/no answer') then this is a drift toward covert communication and pure steganography.

2.2 Semi-structures

When dealing with graphs in general and semi-structures in particular, we are faced with the issue of uniquely identifying and referencing nodes [1]. In graph theory, this is summarized under the term *canonical labeling* [2] [3] [6] [9] and no solution has been provided with a high enough degree of generality.

Thus, before deploying any specific mark encoding techniques we have to ensure a resilient labeling scheme, able to survive minor modifications and attacks on the actual graph structure. We show how content specific watermarking techniques (for node content watermarking) coupled with a technique of content summarization provide a resilient labeling scheme, suited for our watermarking purposes. The value-carrying nodes are solving the labeling issue in quite a surprising manner.

2.3 Attacks

Given a certain value carrying watermarked semi-structure several attack options present themselves, including: elimination of value-"insignificant" nodes (A1), elimination of inter node relations (A2), value preserving graph partitioning into independent usable partitions (A3), modification of node content, within usability vicinity (A4), addition of value insignificant nodes aimed at destroying ulterior labeling attempts (A5). One has to keep in mind the ultimate goal of any attack, namely eliminating the watermark property, while preserving most of the attached value, within usability limits [2].

In order to prevent success for A5, we propose a preliminary step of **value pruning** in which all value-insignificant nodes are marked as to-be-ignored in the ulterior watermarking steps. Another approach deploys structural changes to bring the semi-structure to the limits of the usability space [12], increasing its fragility to further modifications and thus the failure likelihood of any ulterior attempts to attack by adding nodes. A4 mandates the ability of the labeling scheme to depend as little as possible on node content or to provide for a mechanism of detecting altered-content nodes at extraction time. Another possibility of defending against A4 would be to actually alter the main considered nodes toward their allowed fragility limit, such that any further un-knowledgeable changes will fail to provide a usable result. Attack

---

[1] Especially if required to maintain consistency *before* and *after* attacks (e.g. possible structural changes).

[2] Collusion attacks are not discussed in this paper as they are relevant when fingerprinting is deployed. Although we envision extensions of this work for fingerprinting, we are not considering these here.

A3 is one of the most powerful challenges. In order to survive it, meaning that the watermark has to be preserved (maybe in a weaker form) also in the resulting graph's partitions, the watermarking scheme has to consider some form of hierarchical embedding in such a way as to "touch" most of the potential partitions in the graph. The issue becomes more complex if the usability domains of all possible graph partitions are unknown, making it difficult to envision the attacker's "cut". Fortunately, in many cases (see Scenarios) the number of available partitioning schemes that make sense and the associated usability domains are limited. Cases A1 and A2 make it necessary to devise a node labeling scheme that tolerates node and edge elimination while preserving most of the other nodes' labels. This is a must because of the necessity to reference nodes at extraction time. Even if there would exist a working traditional canonical graph labeling algorithm it would need to be heavily modified in order to provide for edge and node removal tolerance. We used the term "heavily" to outline the fact that canonical labeling has always been linked to proofs of graph isomorphism, whereas in this case the trend is aimed exactly toward the opposite, namely preserving node labels in the context of admittedly slight graph changes.

3 Solution

3.1 Tolerant Canonical Labeling

The node labeling scheme is at the heart of watermarking semi-structures. The ability to identify and reference nodes within the to-be-watermarked structure is of paramount importance and the labeling scheme has to take into account the specifics of the case, in particular the requirement to be able to "recognize" all relevant nodes in an attacked version of the graph, based on labels issued on the original one.

Although canonical labeling for graphs was known for a long time to be a hard problem of graph theory, specific algorithms have been developed for some cases . In particular, reasonable solutions have been proposed for tree canonical labeling and apparently, many semi-structure watermarking applications (e.g. HTML) would fit the assumption of tree structuring. One can partition existing value-carrying semi-structures into a set of tree-shapes and remaining structural elements. Watermarking only those partitions might provide enough power and reduce the problem to tree shapes. Unfortunately the requirement of being able to label nodes consistently before and especially *after* attacks, renders useless existing tree canonical labeling algorithms due to their high fragility to any changes (e.g. attacks) made to the structure.

Fortunately, the dual nature of semi-structures enables a novel approach to labeling, the main idea being the use of a combination of structural and node content information.

On the one hand, content is combined in computing a node's label by using a special "tolerant" summary (i.e. a function of the content with specific properties, see Section 3.2) of its content. The assumption here is that content changes are small and that we are able to construct a function of the node content that will basically degrade gracefully with minor alterations to its input. On the other hand some node topology information is necessarily involved in the relative position of the node versus its neighbors and the entire graph. One simple solution that comes to mind is to use the

neighbors' labels, which does capture the position of the current node in relationship to its neighbors, and through the entire labeling scheme, applied recursively, to the graph as a whole. Thus the primitive labeling algorithm can be summarized by the following iterative formula:

$$l(node) = \alpha * l(node) + \gamma * \sum_{nb \in neighbors(node)} l(nb)$$

Note: α determines the "weight" of the node content in the labeling scheme. If essential content changes are unlikely in an attack, α is to be increased so as to provide labeling stability. γ provides control over being able to more specifically localize the node with respect to the neighbors and also to the entire graph. If structural changes are highly unlikely in the course of an attack an increased γ provides for stability [3].

The algorithm starts with the initial labels as being the keyed tolerant content summary values $SUMMARY(key, content(node), key)$ (see section 3.2).

Step One. The first step performs a number of iterations i over the formula above (this number being kept as part of the watermark detection key and used later on in re-labeling the attacked graph), until the necessary labeling provisions are met. At this stage we are mainly concerned with a minimal number of identical labels [4].

Step Two. In order to provide resilience to a certain number of graph modifications ("surgery"), the next step is to artificially degrade the graph and re-perform step one again.

Intuitively (for experimental results see Section 4.2), removing and/or adding nodes and relations to the graph will result in changes in the initial labeling performed on an un-modified graph. Control over those changes is enabled by specifying the α and γ values. Experiments show that, given a graph, for certain α and γ value bounds, labeling becomes controllable.

The result of step two, for each node, is a range of values for the corresponding label, depending also on the three main control factors (step-one iteration number, α, γ). The actual label of the node will be defined by the lower and upper bounds of the resulting labeling range. This basically ensures that, when labeling the attacked/modified version of the graph (i.e. by performing step one of this same algorithm later on, in court), the resulting labels will fall within the corresponding node's label interval with a high likelihood. For a given set of surgeries, performing the labeling algorithm in the space of (α, γ, i) results in a "bounded space of labeling points" (see Figure 1).

The next challenge is to identify an optimum in this "space", given a certain ability to compare two particular "points". Remember that a "point" corresponds to a labeled graph as a set of interval-labels for the graph's nodes, given the particular (α, γ, i) coordinates. Our initial comparison formula for two different graph interval-label

---

[3] It might be interesting to note the fact that if γ is 0, this labeling scheme converges to a simple intuitive content-based addressing scheme.

[4] A number of iterations at least equal to the diameter of the graph are necessary in order to localize a given node with respect to the entire graph. But this is sometimes not desired nor required. The ability to set the number of performed iterations and make it part of the recovery key is another point of control over the labeling scheme.

sets aims at capturing optimality in terms of both minimal number of label overlaps within each set as well as minimal potential for future overlap. If the two considered "points" are the actual interval-label sets $A = \{(a_{11}, a_{12}), ..., (a_{n1}, a_{n2})\}$ and $B = \{(b_{11}, b_{12}), ..., (b_{n1}, b_{n2})\}$ (i.e. (a_{i1}, a_{i2}) is the label-interval corresponding to node i in the graph labeling A) then the comparison formula is

$compare_1(A, B) = $ overlaps(B) $\times$ avg_overlap(B) - overlaps(A) $\times$ avg_overlap(A)

$compare_2(A, B) = $ closest_inter_label_size(A) - closest_inter_label_size(B)

$compare(A, B) = compare_1(A, B) + compare_2(A, B)$

where $overlaps(X)$ is the number of overlapping interval-labels in labeling X, $avg\_overlap(X)$ the average interval size of the overlapping portions and $closest\_inter\_label\_size(X)$ the size of the interval between the closest two interval-labels in X. Intuitively, $compare_1()$ models and compares the current optimality of both labelings and $compare_2()$ captures the potential for future overlap (i.e. because having very "close" interval-labels hints to possible issues in labeling the graph in an attacked version of it).
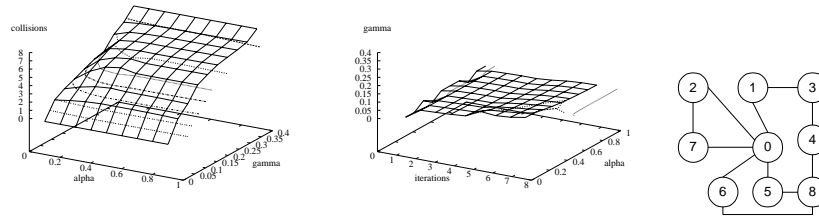
Fig. 1. (a) The surface defining the composite label collisions appearing after 4 stages of training (i.e. $i = 4$) with a random generated set of surgeries applied to the graph. It is to be noted that lower γ values seem to yield a lower number of composite label collisions but in turn results in a lower resistance to structural attacks (i.e. as labeling will not be as resilient to graph surgeries). (b) The zero-collision (for composite labels) surface in the (iterations,alpha,gamma) space corresponding to the same set of surgeries. Its existence proves the ability to label resiliently (to the considered surgeries) without colliding resulting composite labels. Computed using the **wmx.*** package. (c) The considered graph.

What happens if overlapping labeling intervals (i.e. "colliding composite labels") occur ?

- If nodes are in virtually identical/indistinguishable positions and with similar content then this is normal. The nodes are marked as such and treated identical throughout the watermarking process.
- If nodes differ in content but positions are similar, or content is close but positions are different, then variations in α, γ and the content summary key are performed in such a way as to differentiate labels
- If nodes differ in both content and position, changing also the iteration number in step one is required.
- If everything has been done and label intervals are still overlapping we can simply "melt" the labels together and treats the nodes as in case 1 (i.e. identical).

In summary, the labeling process (i) collects all relevant labeling data over a number of iterations in which all of (α, γ, numbers of step-one iterations (i), content summary key and number of performed surgeries) are varied, and then (ii) decides upon a certain point in this space (defined by α, γ, i, content summary key and number of performed surgeries) which minimizes the number of overlapping label intervals and the potential for future overlaps (in case of attacks). By adapting to the given structure (i.e. through adjusting of α, γ, etc), the labeling algorithm allows for control over the required trade-offs between label resilience and tolerated graph changes [5].

```
label(graph G)
  foreach node n do label(n) = SUMMARY(key, n)
  for (α = 0.1; α < 0.9; α = α + 0.1)
    for (γ = 0.1; γ < 0.9; γ = γ + 0.1)
      foreach artificial graph "surgery" (i.e. expected attacks) do
        perform surgery (remove node/relation(s))
        for (iteration = 1; iteration < diameter(G); iteration + +)
          foreach node n do
            label(n) = α × label(n) + γ × ∑_neighbors(n) label(nb)
        foreach node n do store label(n)
      foreach node n do store clabel(n) = [min(label(n)), max(label(n))]
  choose (α, γ) minimizing the number of overlapping label intervals
```

Fig. 2. Labeling Algorithm.

3.2 Tolerant Content Summaries

Finding an appropriate (set of) content summary function(s) that satisfy the requirements above is not trivial and strongly tied to the type of semi-structure node content and its associated transforms and envisioned attacks. The main requirements of the content summary functions considered are *the ability to be at the same time quite content specific while also degrading gracefully with minor changes in the content*. The idea is to capture and quantify certain global specific properties of the content that are still preserved in the watermarking/attack process. Research by Ari Juels et. al. [7] investigates a related notion, suggestively qualified as "fuzzy commitment".

In our implementation we used a simple content summary, a linear combination of the high order bits of the node content. The assumption here was that high order bits are reasonably stable to changes and attacks. Other applications require different consideration. In the case of JPEG content for example, frequency domain transforms like the DCT could be considered. The tolerant summary of a JPEG file would be a combination of the most significant bits of its significant DCT coefficients etc.

---

[5] Special consideration needs to be offered to the case of an attack modifying all existing nodes' content in a similar fashion. Alteration to the labeling scheme can be prevented in this case by introducing an additional final step of globally normalizing the labels (label intervals).

Feature extraction algorithms (e.g. property histograms) should be also investigated for multimedia content as a means to provide a tolerant content summary.

3.3 Watermarking Algorithm

The main idea behind our algorithm is to use the structural resilience of the labeling scheme while leveraging content-specific one-bit watermarking methods for each node. In other words, each node in the semi-structure is considered to be a potential recipient of a one-bit watermark (using a traditional content-type specific marking method), while the actual instances of these encodings are going to be determined in a secret fashion by the node labels.

Let $clabels()$ be the composite labeling intervals as computed above (see Section 3.1). Depending on the size of the intervals in $clabels()$, choose b as the maximal number of most significant bits that can be considered in the numbers of every interval such that $\forall (x,y)_j \in clabels(), msb(x,b) = msb(y,b)$, where $msb(x,b)$ are the most significant b bits of x. In other words we aim to discover an interval-specific invariant. For each node j and corresponding interval $(x,y)_j$ by notation, let $msb_j = msb(x,b)$.

Let k be a seed to a b-bit random number generator RND and $k_1, ..., k_n$ the first n b-bit random numbers produced by RND after a secret initial warm-up run. We say that node j is "fit" for encoding iff $(msb_j \oplus k_j) \bmod e = 0$, where e is a adjustable encoding parameter determining the percentage of considered nodes. In other words, a node is considered "fit" if its label satisfies a certain secret criteria. On the one hand this ensures the secrecy and resilience of our method, on the other hand, it effectively "modulates" the watermark encoding process according to the actual graph structure. This naturally provides a witness and rights "protector" for the structure itself.

---

embed(G, wm, k, e)
  $clabels() = \textbf{label}(\text{graph})$
  $b = \{max(z)|\forall (x,y)_j \in clabels(), msb(x,z) = msb(y,z)\}$
  initialize $RND(k)$
  $i = 0$
  sort $clabels()$
  foreach $(x,y)_k \in clabels()$ **do**
    $k_j = RND()$
    if $((msb(x,b) \oplus k_j) \bmod e = 0)$ **then**
      content_wm_node$(k,(wm_i \oplus lsb(k_j, 1)),k_j)$
      $i = i + 1$

detect(G, k, e, b)
  $clabels() = \textbf{label}(\text{graph})$
  initialize $RND(k)$
  $i = 0$
  sort $clabels()$
  foreach $(x,y)_k \in clabels()$ **do**
    $k_j = RND()$
    if $((msb(x,b) \oplus k_j) \bmod e = 0)$ **then**
      $wm_i = \textbf{content\_det\_node}(k,k_j)$
      $i = i + 1$

Fig. 3. (a) Watermark Embedding Algorithm (b) Watermark Detection Algorithm

Each node considered fit is then watermarked with the one-bit watermark defined by the XOR between the least significant bit of its corresponding k_j and $wm_i, i \in (0, |wm|)$ the corresponding watermark bit. Because of the e factor we have an average guaranteed bandwidth of $\frac{n}{e}$. In case the watermark length $|wm|$ is less than $\frac{n}{e}$, we can choose for example, the watermark bit $(\frac{n}{e} \bmod |wm|)$, effectively deploying a majority voting scheme etc. The 1-bit watermark embedding uses traditional (node) content-watermarking techniques. Because these do not constitute the main contribution of

this research, in our abstract watermarking suite we considered a simple place-holder, in which each node contains a large integer value. A "1" watermark bit is considered to be present when the value is odd, a "0" otherwise.

Note: The key used in the content-watermarking technique can be the same k_j or any other agreed upon secret. There might be some benefit associated with using the *same* single key for all nodes as this could defeat inter-node collusion attacks (in which the same node content is watermarked with different keys). It is also assumed that the content watermarking method deployed is respecting the maximum allowable distortion bounds associated with the given content. In particular, these node content-specific constraints are not impacting structural consistency. In other words, slight modifications to the actual node content (e.g. JPEG images or natural language text) do not alter global structural consistency constraints. This is subject to further research.

In the decoding phase, the $clabels()$ set is re-computed. The result should be identical (in case no alterations occurred) or fairly close (because of the inherent labeling tolerance to alterations). We know $k_1, ..., k_n$, the secret node-selection keys and b. Based on these values and the composite labels, the algorithm performs node-selection and identifies a majority (or all in the case of some graph alterations occurring) of the initial nodes that were watermarked. Content-specific watermark detection is then applied to each node to retrieve each watermark bit.

In order to perform error correction, if enough bandwidth is available (e.g. n is large enough), the algorithm embeds multiple copies of the watermark (or any other error correction encoding). Upon detection, majority voting is deployed to increase the likelihood of accurate detection.

3.4   Discussion

What happens if we cannot discover a "nice enough" b ? That is, what happens if different label intervals in $clabels()$ are behaving so "wildly" apart that b is going to be really small. In other words, what if there exists a node whose composite label interval has its endpoints very very far away such that the MSB common bits are just a few, or even none.

We would argue that this is a highly unlikely scenario and experiments confirm it. But if it is indeed the case then we have several options, one of which is simply ignoring the label(s) that are having far-away endpoints. Another option would be to introduce an initial normalizing step in which all the labels are normalized with respect to a common "average" value (e.g. means of means).

In order to fight false-positive claims in court we ask: What is the probability of a given watermark of length m to be detected in a random graph of size n. The assumption is of course that $m < \frac{n}{e}$. It is easy to prove that this probability is $(\frac{1}{2})^m$. In case multiple embeddings are used (e.g. majority voting) and all available bits are utilized, this probability decreases even more to $(\frac{1}{2})^{\frac{n}{e}}$. For example, in the case of a structure with 60 nodes and with $e = 3$, this probability reads *one in a million*, reasonably low.

In the absence of additional information, Mallory, faced with the issue of destroying the watermark while preserving the value of the data, has only one alternative

available, namely a random attack. Two sub-types of attacks present themselves as outlined in Section 2.3: structural and node-content altering.

Structural attacks are handled by the tolerant nature of the labeling scheme and an experimental analysis is presented in Section 4.2. Here we are concerned with the node-content alteration attacks. We ask: what is the probability of success of such an attack ? In other words, if an attacker starts to randomly alter a total number of a nodes and succeeds in each case to flip the embedded watermark bit with a success rate p, what is the probability of success of altering at least $r, r < a$ watermark bits in the result, $P(r, a)$? It can be shown that $P(r, a) = \sum_{i=r}^{a} [aCi] \times p^a \times (1-p)^{a-i}$.

Now, remember that only every e-th node is watermarked, thus the attacker effectively attacks only an average of $\frac{a}{e}$ nodes actually watermarked. If $r > \frac{a}{e}$ then $P(r, a) = 0$. In the case of $r < \frac{a}{e}$ we have we have the corrected version

$$P(r, a) = \sum_{i=r}^{(\frac{a}{e})} [(\frac{a}{e})Ci] \times p^{(\frac{a}{e})} \times (1-p)^{(\frac{a}{e})-i}$$

If $r = 4$, $p = 10\%$, $a = 20$ (33% of the nodes are altered by the attacker !) and $e = 4$, we have $P(4, 20) \approx 55 \times 10^{-6}$, again a reasonable figure, reading *fifty-five in a million*. Space constraints do not allow for a more in-depth analysis.

4   Implementation and Experiments

4.1   The wmx.* package

wmx.* is our java software test-bed package for watermarking abstract semi-structures. We developed and implemented the algorithms presented and experimented with various semi-structured shapes. The package allows for dynamic generation and storing of graphs, graph surgeries and attacks, as well as for runtime customizable labeling and watermarking parameters (e.g. α, γ, *iterations*, *collision_bounds*). In the experiments, most of the nodes were defined as allowing for specific node content watermarking that encodes one bit per node.

Given the low probability of attack and false-positives discussed above in Section 3.4, one thing we believe needs to be analyzed in more detail is the actual feasibility and resilience of the labeling method. Given its importance to the overall algorithm, we implemented a test suite that allows experiments on abstract, dynamically redefine-able structures composed of a customizable number of nodes with associated random generated (or predefined) content. We then extended the package to allow for watermarking of abstract define-able semi-structures. We performed experiments on structures with varying number of nodes and levels of connectedness. The computations were conducted on a 500Mhz PC with 128MB RAM running Linux. Code was written in Java.

4.2   Experiments

One of our main concern was labeling collisions, i.e. composite label sets *clabels*() in which multiple labeling intervals are overlapping. These appear as a result of the

training surgery phase, in which modifications are performed to the graph to produce the new label set. It is bad news as it creates potential ambiguity in the detection process. Surprisingly, in most cases, by adjusting the labeling training parameters $\alpha, \gamma, iterations$ we could obtain points that did feature zero collisions. In Figure 4 we show the zero-collision surfaces (in the α, γ space, with 3 training iterations) for two simple structures.

The considered set of training surgeries (i.e. the set of surgeries performed on the original graph before each individual labeling iteration) was randomly computer-generated from a set of global surgeries and included periferic node removals, edge additions and removals. (To be noted that this is consistent with the assumptions made in section 2.3 when discussing attack A5).
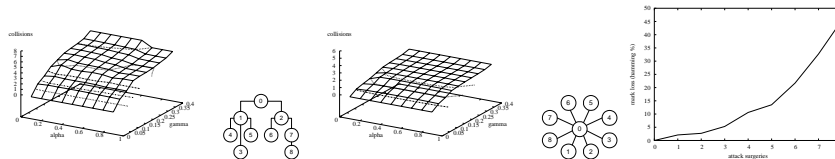
Fig. 4. Surfaces defining the composite label collisions appearing after 3 stages of training with a random generated set of surgeries. (a) Tree shaped graph. Much of the web content online is tree-shaped. Again, note that lower γ values seem to yield a lower number of composite label collisions, with drawbacks as presented in figure 1 (b). (b) Star shaped graph. Note the smoother shape and the lower collision bounds, compared to (a). The same nodes were used, differently interconnected. Computed using the **wmx.*** package. (c) Averaged watermark loss over 10 runs of an 8 bit watermark embedded into an arbitrary 32 node graph with 64 edges. Surgery attacks are applied randomly (node removals 60%, link addition 20%, link removal 20%). The labeling scheme was trained for 3 surgeries.

In Figure 4 (c) we show the watermark behavior in the case of a random artificially generated structure with 32 nodes and 64 edges. The embedded watermark is 8 bits long. The labeling scheme was trained for 3 surgeries, also $e = 3$ (average bandwidth available is thus 10.6 bits, enough for the 8 bit watermark). It can be seen how composite labeling training results in highly resilient labels. As the number of attack surgeries increases, the watermark degrades slightly. The results are averaged over 10 runs on the same graph with different random attacks. When 8 attack surgeries are applied to the graph we can still recover 60-65% of the watermark. One has to consider also the fact that an attacker is bound not to modify the structure too much as it will eventually distort.

5   Conclusions

We introduced an algorithm for rights protection watermarking of semi-structured content. More specifically we are concerned with protecting the value inherent in the

structure itself. Various new challenges are associated with this new domain. Benefiting from the dual nature of semi-structures, our algorithm makes use of both the available node content as well as of the value-carrying structure, through the processes of canonical labeling, node content summarization and content-specific mark encoding. The idea behind content-specific mark encoding is to use traditional known watermarking techniques, in encoding parts of the watermark in the node content. Providing a canonical labeling scheme, "trained" to tolerance for a set of graph modifications is essential in being able to later-on identify nodes selected in the 1-bit node mark content-specific encoding process. Our algorithm does not require the original un-watermarked object in order to perform mark detection. Further work is required in improving content summarization and tolerant labeling. Different application domains will require specific approaches. An alternative idea would be using bandwidth available in the specifications of inter-node relations.

References

1. M.J. Atallah, V. Raskin, C. F. Hempelmann, M. Karahan, R. Sion, K. E. Triezenberg, and U. Topkara. Natural language watermarking and tamperproofing. In *Lecture Notes in Computer Science, Proc. 5th International Information Hiding Workshop 2002*. Springer Verlag, 2002.
2. L. Babai and L. Kucera. Canonical labeling of graphs in linear average time. In *Proc. 20th IEEE Symposium on Foundations of Computer Science, 39-46.*, 1979.
3. L. Babai and E. Luks. Canonical labeling of graphs. In *Fifteenth Annual ACM Symposium on Theory of Computing, pages 171–183. ACM*, 1983.
4. Christian Collberg and Clark Thomborson. Software watermarking: Models and dynamic embeddings. In *Principles of Programming Languages*, San Antonio, TX, January 1999.
5. I. Cox, J. Bloom, and M. Miller. Digital watermarking. In *Digital Watermarking*. Morgan Kaufmann, 2001.
6. Faulon J. Automorphism partitioning and canonical labeling can be solved in polynomial time for molecular graphs. In *J. Chem. Inf. Comput. Sci. 38, 1998, 432-444.*, 1998.
7. Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *ACM Conference on Computer and Communications Security*, pages 28–36, 1999.
8. S. Katzenbeisser and F. Petitcolas (editors). Information hiding techniques for steganography and digital watermarking. Artech House, 2001.
9. Ludek Kucera. Canonical labeling of regular graphs in linear average time. In *IEEE Symposium on Foundations of Computer Science*, pages 271–279, 1987.
10. J. Palsberg, S. Krishnaswamy, M. Kwon, D. Ma, Q. Shao, and Y. Zhang. Experience with software watermarking. In *Proceedings of ACSAC, 16th Annual Computer Security Applications Conference*, pages 308–316, 2000.
11. Radu Sion. Proving ownership over categorical data. In *Proceedings of the IEEE International Conference on Data Engineering ICDE*, 2004.
12. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Power: Metrics for evaluating watermarking algorithms. In *Proceedings of IEEE ITCC 2002*. IEEE Computer Society Press, 2002.
13. Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Rights protection for relational data. In *Proceedings of ACM SIGMOD*, 2003.