Watermarking Road Maps against Crop and Merge Attacks

Kai Jiang   Kenny Q. Zhu
Shanghai Jiao Tong University
Shanghai, China
jkai@sjtu.edu.cn,
kzhu@cs.sjtu.edu.cn

Yan Huang
University of North Texas
Denton, TX, USA
huangyan@unt.edu

Xiaobin Ma
1 Oracle Drive
Nashua, NH, USA
xiaobin@cs.umn.edu

ABSTRACT

Past research on watermarking digital road maps has been focused on deterring common attacks such as adding noise to the whole map so as to destroy the embedded watermarks. This paper focuses on two less common but increasingly used types of attack: crop attacks and merge attacks. Crop attack crops a fragment of the original map and uses the fragment as a new map. When the new map is much smaller than the original map, it is called massive cropping. Merge attack merges maps from various sources together to form a new map. Conventional watermarking techniques fail against these attacks either because they require global information from the whole map or they must add too many local watermarks and affect the usability of the maps. This paper proposes a novel quad-tree based blind watermarking scheme that partitions the original map according to the quad-tree and plants just one single bit in each sub-region of the map. The approach achieves almost 100% detection accuracy for moderate crop and merge attacks, and over 80% accuracy with more than 95% of the original map cropped and removed. Furthermore, the method introduces very little distortion to the original map: to effectively protect a 23.5MB Minneapolis-St.Paul map against crop and merge as well as other common attacks, only 423 bits or 53 bytes of watermark is required. [1]

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]: Information hiding; D.4.6 [**Security and Protection**]: Authentication

Keywords

Watermark, digital road map, quad-tree, crop attack, merge attack

1. INTRODUCTION

Digital watermarking is an important technique to protect the copyrights of digital products such as images, audios and videos. A watermark is small amount of digital noise embedded into the digital representation of the products. Watermarking is different from encryption, another method used to protect digital content from unauthorized access. Watermarks embedded in a product should not be *perceived* by the user or the application and hence don't affect the normal use of the product, whereas encrypted product cannot be used unless the user has the means to decrypt the product, usually with the help of a secret key. Once the content is decrypted, unlimited illegal copies of the product can be made and used as if they were legal copies. Digital watermarking complements encryption. By embedding some watermarks that are hard to remove, one can always claim the ownership of the product after detecting the watermarks.

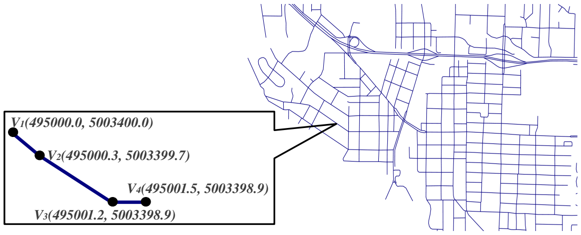

V_1(495000.0, 5003400.0)
V_2(495000.3, 5003399.7)
V_4(495001.5, 5003398.9)
V_3(495001.2, 5003398.9)

Figure 1: A Digital Road Map for Part of Anoka County, MN

In this paper, we are concerned with the protection of copyrights of digital road maps by watermarking. A digital road map is a vector graph representation of roads in a geographical region. Such maps are widely used in Geographic Positioning Systems (GPS) and other GIS or location-based applications. Figure 1 shows an example of a snippet of a road map of Anoka County, MN in the United States. In this map, a road is represented by a *polyline*, shown in the blow-up image, where a polyline is a sequence of connected straight line segments. Each line segment is presented digitally by its two end vertexes in terms of (x, y) coordinates, where x and y are latitude and longitude of the point on earth, respectively. Very wide two-way roads (e.g., freeways) with central dividers are represented by two (often) parallel polylines, which are also shown in Figure 1.

The standard watermarking framework for digital road maps (shown in Figure 2) also adopts a two-step approach. The two key modules in the framework are the watermark insertion and detection algorithms. These two modules typically share some common secret keys which are only known to the algorithms. Some detection algorithms require the original map while others don't. When digital road maps are unlawfully copied for commercial use, they often undergo various modifications or transformations in hope that the original watermarks are either removed or become undetectable. Such modifications to the map is called *attacks* on the watermarks. Existing attacks include attempts to remove or alter the watermarks,

[1] Kenny Q. Zhu (the corresponding author) is partially supported by NSFC grants 61033002 and 61100050.

IH&MMSec'13, June 17–19, 2013, Montpellier, France.

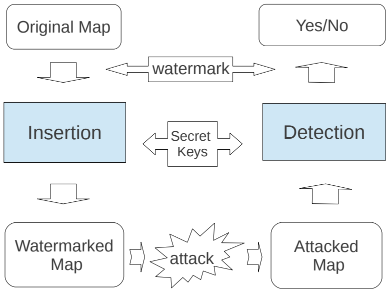


Figure 2: General Watermarking Framework For Road Maps

adding more noises to the map, cutting a map into smaller pieces or merging multiple maps from different sources. The last two attacks, which we call "*crop attack*" and "*merge attack*" are less common, and they are the focus of our research in this paper.

A scenario of a crop attack goes like this: an attacker obtains a big map, e.g., the map of the state of Minnesota, and crops out the Minneapolis - St.Paul area to create a "new" twin-city map. In the case of a merge attack, the attacker extracts maps for various counties in Minnesota from several different digital maps and composes a new Minnesota map by aligning these sub-maps properly together. The first attack is easier to implement, while the second attack is more difficult to carry out in practice and harder to defeat.

Existing watermarking techniques come in two categories, *global watermarking* and *local watermarking*. In global watermarking, the insertion module computes an overall watermark based on the global information of the data, and inserts this watermark over the whole data. Detection requires the reconstruction of the original watermark which requires global information as well. As such, these techniques cannot handle crop or merge attacks.

In local watermarking, the insertion module generates many watermarks for different regions of the data, and each watermark is computed from the local information of the corresponding regions. For example, one can partition a map into many smaller regions, and insert a watermark according to the properties of the data in each region. Local road map watermarking schemes [15, 16, 23] have the potential of surviving crop attacks. However, to handle "massive cropping", i.e., cropping out a very small piece of the original map, existing local schemes have to resort to fine-grained partitioning and the insertion of many more watermarks which may affect the perception of the map. Moreover, many of these techniques require coordination between the detection of watermarks in adjacent sub-regions (e.g., some repetitive patterns) to conclude the authenticity of the overall map. Such coordination may fail if it happens at the boundary of the cropping. Such massive cropping attacks are real, since in our previous example, the Twin-Cities area is much smaller than the whole state of Minnesota.

None of the existing local methods can defeat the merge attack because all of them compute a global confidence score based on the watermarks extracted from individual sub-regions. If a significant part of the map comes from a foreign source without the inserted watermarks, these methods typically give a low confidence score and fail to identify part of the map as being authentic. Without a proper data structure, these methods also suffer from high computation complexity if they attempt to "guess" where the watermarked region is in the map.

In this paper, we propose a novel local watermarking approach which partitions the original map by a modified quad-tree structure. The depth of the tree is determined by the road density in the region. We compute the local watermark for each partition by the total length of the roads in that partition and each watermark is represented by *one bit* change in the original data. During detection phase, our approach can reconstruct the quad-tree and identify the potential locations of the embedded watermarks, and can report with high confidence if a given attacked map (be it traditional attacks or crop/merge attacks) contains a sub-region which carries the original watermarks as well as reporting that sub-region itself. In addition, the detection method in our approach doesn't require the original map, but only needs a "secret grid boundary" which serves as the key. The size of this key is negligible compared to the original data. Watermark detection scheme without the need for original data is also known as "blind watermarking."

This paper makes the following key contributions:

- *The framework defeats massive cropping and merging attacks with high accuracy.* Our experiments show that our approach achieves 100% detection accuracy for moderate cropping and more than 80% accuracy with more than 95% of the original map cropped, whereas the accuracy of two other state-of-the-art approaches degrades to around 20% with similar massive cropping. Our method outperforms the peers by similar margins under merge attacks.

- *The framework incurs little distortion.* The map is partitioned into sub-regions of various sizes *on demand*. Watermarking position is calculated by information inside these sub-regions. We use a one-bit watermark to represent that the sub-region is watermarked. Consequently, to effectively watermark the whole Twin-Cities 7-counties map (23.5MB), our approach merely modified 423 *bits*.

- *The framework is lightweight.* It is a blind watermarking method which requires no original map for watermark detection, and the time complexity of both the insertion and detection algorithm is $O(|V|log\mathcal{L})$ where $|V|$ is the number of vertices in the map and $\mathcal{L}$ is the total length of roads in the map.

The remainder of this paper is organized as follows. Section 2 introduces some preliminaries about the watermarking GIS digital data and formalizes the problems of interest. Section 3 discusses the proposed digital watermarking approach in detail. Section 4 describes the experiment setup and presents evaluation results on a real digital map data set. Section 5 discusses the related work, and we conclude the paper with some further work in section 6.

2. PROBLEM DEFINITION

In this section, we first define digital road maps and distortions to the map. We then give the problem definition of watermarking digital road maps in terms of the interfaces of two functions, *insertion* and *detection*. Finally we describe some common attacks on digital maps with special emphasis on the two difficult attacks: crop and merge attacks.

2.1 Digital Road Map and Its Distortions

A digital road map, M, is a *view* of a graph G with a set of vertices V and a set of edges E. M consists of a set of k roads R_i, where each road is a *polyline* represented by a sequence of vertices in the form of (x, y) coordinates in a geographical coordinate system (e.g., latitudes and longitudes). Formally,

$$\begin{aligned} G &= \{V, E\} \\ M &= \bigcup_{i=1}^{k} R_i, \text{where } k \geq 1; \text{ and} \\ R_i &= [V_1,\ V_2, \ldots, V_m] \\ &\quad \text{where } m \leq |V|; \forall i \neq j:\ V_i \neq V_j; \\ &\quad \forall i \in [1, m]:\ (V_i, V_{i+1}) \in E; \text{ and} \\ &\quad \forall i \in [1, m]:\ V_i = (x_i, y_i). \end{aligned}$$

Whether we are watermarking or attacking a given map, we are essentially changing the original map, or adding distortion. In order to maintain usability of the map, the amount of distortion must not be larger than a threshold η known as *perception tolerance*. We consider three types of distortion one can make to a map: *perturb* some vertices, *insert* some new vertices into a road, or *delete* some vertices from a road.

Given a road R, and a changed road R', let P be the set of nodes in R which are perturbed, and I be the set of nodes in R' which are newly inserted, and D be the set in R of nodes which are deleted, the distortion between R and R' is defined as

$$\delta(R, R') = \delta(P) + \delta(I) + \delta(D)$$

where

$$\begin{aligned} \delta(P) &= \sum_{V_i \in P} ||V_i' - V_i|| \\ \delta(I) &= \sum_{V_i \in I} d(V_i, R) \\ \delta(D) &= \sum_{V_i \in D} d(V_i, R') \end{aligned}$$

and

$$d(V_i, R) = \begin{cases} ||V_i - V_{i+1}|| & \text{if } i = 1,\ \{V_i, V_{i+1}\} \subseteq R \\ ||V_i - V_{i-1}|| & \text{if } i = |R|,\ \{V_i, V_{i-1}\} \subseteq R \\ |||\boldsymbol{\alpha}|| \sin \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle| & \text{if } \{V_{i-1}, V_i, V_{i+1}\} \subseteq R \end{cases}$$

where $\boldsymbol{\alpha} = V_i - V_{i-1}$ and $\boldsymbol{\beta} = V_{i+1} - V_{i-1}$, and $\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle$ means the angle of intersection between vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Here $d(V_i, R)$(see Figure 3) means the distance between a node V_i and the line segment (V_{i-1}, V_{i+1}), or if V_i happens to be the end of a road, the distance to its adjacent node in the road.

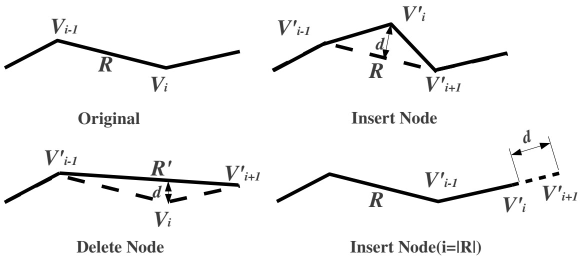


Figure 3: Distance Measurement function d(V, R)

The distortion to the whole map is hence the total distortion to all its roads, normalized by the total lengths of all roads:

$$\delta(M, M') = \frac{\sum_{R \in M, R' \in M'} \delta(R, R')}{\sum_{R \in M} length(R)}.$$

In the remainder of this paper, we use δ_w to represent the $\delta(M, M')$ caused by watermarking, and use δ_n for $\delta(M, M')$ due to noise.

2.2 Insertion and Detection of Watermarks

For a digital road map M, watermark W, and some secret key K (to enhance robustness against attacks), the interface of watermarking process can be represented by two functions $\mathcal{I}$ and $\mathcal{D}$.

$$\begin{aligned} \mathcal{I} &: (M,\ W,\ K) \rightarrow M' \\ \mathcal{D} &: (M,\ W,\ K) \rightarrow \{T \mid F\} \end{aligned}$$

where M' is the watermarked map, $\delta(M, M') \leq \eta$, and detection process returns a boolean value of either *true* or *false*.

2.3 Attacks

There are many different attacks on digital road map. In this section We focus on into three types: the first type is a common attack which introduces random noises into an illegal copy of the map. The other two types are more complex attacks which involves cropping a given map into smaller pieces and merging pieces together from various sources.

Noise Attack

An attacker disturbs the watermarks in a map by selecting a subset of the nodes in the map and perturb the position of these nodes slightly. The subset could be small compared to the whole map:

$$noise(M) = \{noise(R) \mid R \in M_1\} \cup M_2$$

where $M_1 \cap M_2 = \emptyset$, $M_1 \cup M_2 = M$, $noise(R) = [perturb(V) \mid V \in R]$, where $[f(V) \mid V \in R]$ a list comprehension constructed from another list R.

Crop Attack

An attacker crop a geographical region from the watermarked map and use it as if it's a new map. The crop attack can defeat almost all global watermarking techniques because the global information can be destroyed by the cropping. Even for many local watermarking techniques, the resistance against cropping is limited if just a small piece of the map is cropped. We define cropping attack as:

$$Crop(M) = \{subseg(R) \mid R \in M'\}$$

where $M' \subseteq M$ and $subseq(R)$ returns a subsequence of road R.

If M' is a very small subset of M, we call the attack "massive crop" attack. None of the existing watermarking approaches can handle massive crop attack.

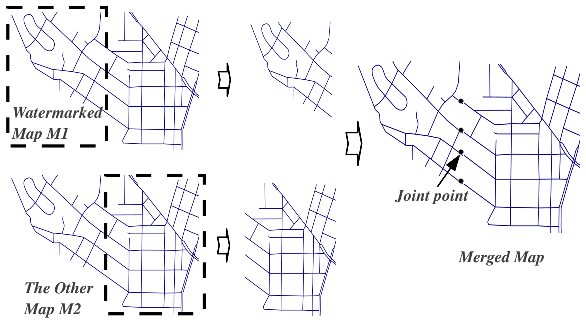


Figure 4: Merge Attack

Merge Attack

This attack involves the merging of multiple maps from different sources (see Figure 4). Maps from various sources differ by either precision of measurement, the granularity of road segmentation, or minor details of roads due to changes to the road over time. However, maps of the same region will be largely identical. This enables the attacker to crop parts from different maps and piece them together to make a new map. There might some slight distortion or inaccuracy at the boundaries but the resulting map is usable. We define the "merge" of two maps as:

$$Merge(M_1, M_2) = M_{sep} \cup M_{con}$$

where

$$\begin{aligned} M_{sep} &= \{R_1 \mid R_1 \in M_1 \text{ and} \\ &\qquad \forall R_2 \in M_2 : connect(R_1, R_2) = false\} \\ \cup &\quad \{R_2 \mid R_2 \in M_2 \text{ and} \\ &\qquad \forall R_1 \in M_1 : connect(R_1, R_2) = false\} \\ M_{con} &= \{join(R_1, R_2) \mid \forall R_1 \in M_1, R_2 \in M_2, \\ &\qquad connect(R_1, R_2) = true\} \end{aligned}$$

and $connect(R_1, R_2)$ is a boolean function that evaluates to true if two roads R_1 and R_2 share a name and an endpoint. $join(R_1, R_2)$ returns a road that joins two connected roads with identical name. M_{sep} and M_{con} are two sets of roads that make up the merged map, where M_{sep} denotes those roads from the original maps which are disjoint and M_{con} denotes a set of roads from the boundaries of the two input maps which are joined together.

3. OUR APPROACH

In this section, we propose a novel watermarking approach, which can effectively survive "massive crop" and "merge" attacks. Our watermarking method inserts negligible watermarks (one-bit only) to locations which are determined by a spatial partitioning algorithm. Then during detection time, the same spatial partitioning algorithm is used with the help of a secret key, which ensures that the resulting segmentation of the space and the input map is almost identical to that of the watermarked map, even though no original map is available. In the following, we first introduce the secret keys used in this framework, then present the space partitioning algorithm, watermark insertion and detection algorithms, before giving analysis of the algorithms. All symbols used in the algorithms are summarized in Table 1.

Table 1: All Symbols Used in Algorithms

Symbol	Definition	Symbol	Definition
G	master grid	PO	sub-region list
M	original data-set	T	MQtree
$\mathcal{R}$	partitioned region	l	secret square size
θ	road length threshold	k	secret hash key

3.1 Secret Keys

The secret keys, often randomly generated, are used to decide where to insert the watermarks. In the proposed algorithm, we use three secret keys: a master grid, a secret minimum bounding rectangle (MBR) and a secret square size.

The *master grid* G is a secret coordinate system which is only known to the map producer. The grid has an origin which is certain position on earth with precise latitude and longitude, and it has a step size which defines the granularity of the grid (see Figure 5):

$$G = \{Origin(x_0, y_0), Step\}$$

Given an original map to watermark, it can be laid out on the master grid, according to the coordinates of the vertices in the map. The *secret MBR* is then the smallest rectangle which coincides with the grid lines and completely encloses the whole map. The red rectangle in Figure 5 is one such secret MBR of the blue road map. In our algorithm, we partition the space according to a modified Quadtree. The MBR of the map serves as an initial of the partition process, which will improve the robustness of our algorithm.

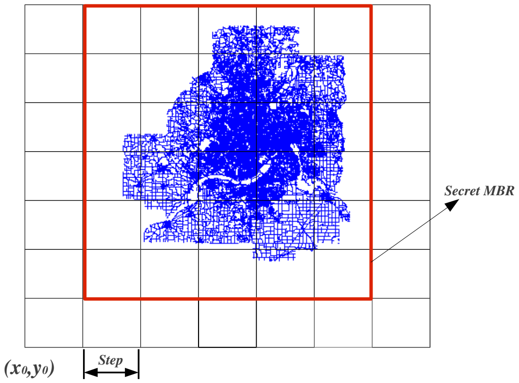
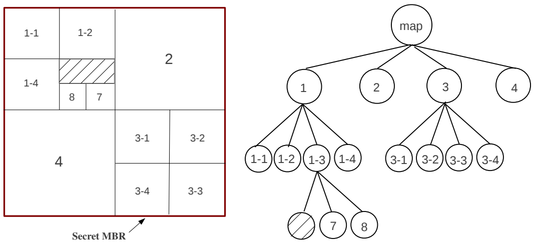


Figure 5: Master Grid and MBR

The *secret square size* is an integer number l that determines the size of a square box that we used to select a small neighborhood of road segments from which to compute the watermarks. More details of the use of l will be presented in Section 3.3.

3.2 Space Partitioning Algorithm

Algorithm 1 is used in both insertion and detection of the watermarks. It recursively partitions the space bounded by the secret MBR for a map into small regions using a modified quad-tree structure (known as MQtree) according to the density of the roads. Each node in MQtree represents a sub-region of the map. A leaf node represents a region in which the total length of road segments within is between θ and 4θ meters. Figure 6(b) illustrate a MQtree created from partitioning an MBR shown in Figure 6(a).



(a) Map
(b) MQtree

Figure 6: MQtree

In Algorithm 1, the secret MBR is generated from the master grid(see Figure 5). Then the map area is partitioned iteratively according to the density of roads such that finally the total road length included in every subregion has more than m meters and less than $4 * \theta$ meters. Finally, we output all of these subregions. Here we modify Quadtree by merging one subregion with less roads with its smallest neighbor when the road length of it is less than θ. Road length of an area cannot be changed arbitrarily due to the constraints of perception tolerance which must be observed by both the watermark producer and the attacker. No matter a watermarked map is cropped or merged with other parts from different maps, this method still produces almost the identical partition results for the watermarked part.

Algorithm 1 Partition

```
Input: $G, M, T, \theta$
Output: $PO, T$
function PARTITION($G,\ M,\ T,\ \theta$)
    $L|PO|T \leftarrow$new queue|new list|$MBR(G,\ M)$
    $L.Push(T)$
    while $L$ is not empty do
        partition region $\mathcal{R} \leftarrow L.pop$ into $\mathcal{R}_i$, $i \in \{1,2,3,4\}$
        for each $\mathcal{R}_i$ do
            if road length in $\mathcal{R}_i < \theta$ then
                merge $\mathcal{R}_i$ with its smaller neighbor
            else if road length in $\mathcal{R}_i > 4\theta$ then
                $\mathcal{R}.children_i \leftarrow \mathcal{R}_i$
                $L.push(\mathcal{R}.children_i)$
            else
                $\mathcal{R}.children_i \leftarrow \mathcal{R}_i$
                insert $\mathcal{R}_i$ to $PO$
return $PO$
```

3.3 Watermark Insertion Algorithm

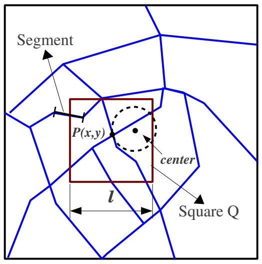


Figure 7: Insertion Strategy

Algorithm 2 first generates a set of sub-regions by partitioning. Then it inserts one watermark into each sub-region at the leaf nodes of the MQtree. The watermark is inserted into a road vertex $P(x, y)$ which is closest to the center of the sub-region (marked by the black box in Figure 7). To compute the exact watermark to be inserted, the algorithm then draw a square box of size l (the other secret key) centered at $P(x, y)$ (the red box in Figure 7). Let sl be the length of road segments that intersects the square box, and let j be a value hashed from sl. We then set the j^{th} least significant bit in x coordinate of P to "1". The hash function in this algorithm must guarantee to hash to the same value before and after watermarking. Even if the watermarked map is attacked (i.e., some end points of those segments intersecting Q are altered), the hash function must still hash to the same j. To this end, we design this function with a tolerance to possible distortions. Assuming that the biggest possible distortion of a single end point is the change of J_{max} least significant bits of the coordinates, then

$$j \quad = \quad Hash(Trans(sl), k)$$

$$\text{where} \quad Trans(x) = x \wedge \underbrace{11\ldots1\overbrace{00\ldots0}^{J_{max}}}_{bit\_length(x)}$$

Algorithm 2 Insert Watermark into Map

```
Input: $G, M, l, k, \theta$
Output: $M$
procedure INSERTION($M,\ G,\ l,\ k,\ \theta$)
    $PO \leftarrow$ PARTITION( $G,\ M,\ NULL,\ \theta$ )
    for each region $\mathcal{R}_i \in PO$ do
        if road length in $\mathcal{R}_i > \theta$ then
            $P \leftarrow$ point closest to $\mathcal{R}_i.center$
            draw a square $Q$ of side length $l$ centered at $P$
            $sl \leftarrow$ length of line segments intersecting $Q$
            j $\leftarrow$ Hash( $k, sl$ )
            $j^{th}$ LSB of $P.x \leftarrow 1$
```

3.4 Watermark Detection Algorithm

In Algorithm 3, we partition the map using the same strategy as insertion. Then we select the data points closest to the center of the regions to detect whether the watermark exists there. Each sub-region which is detected to contain a watermark casts a vote which collectively contributes to the final decision of whether a larger area is watermarked as a whole. For leaf nodes of the MQtree, if the bit value at the right position is "1", we mark this sub-region as a "match". Function STATS(T) (Algorithm 4) computes two statistics for each node of the MQtree: the total number leaf nodes underneath the node (T.total) and the total number of nodes which has been marked as "match" (T.match). If the detection confidence $conf(T_i)$ is larger than a threshold ξ for for any non-leaf node T_i, the watermark in the map is successfully identified. We define the detection confidence of T as

$$conf(T) = 1 - \sum_{i=n}^{N} \binom{N}{i} \left(\frac{1}{2}\right)^{N-i} \left(\frac{1}{2}\right)^{i} \qquad (1)$$

where N is T.total and n is T.match. In our approach, the data points where we select to insert watermark may already contain "1" at the specified bit position. Here we assume uniform distribution and hence the probability of "1" being already present is 1/2.

Algorithm 3 Detect Watermark from a Suspicious Map

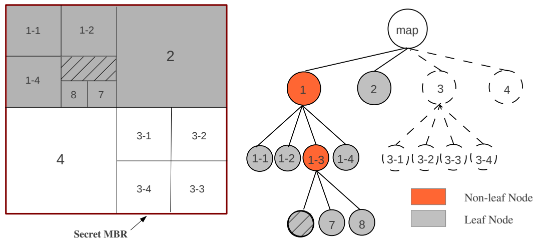
```
Input: $G, M, l, k, \theta$
Output: $Yes/No$
function DETECTION($M,\ G,\ l,\ k,\ \theta$)
    $T \leftarrow$ new MQtree
    $PO \leftarrow$ PARTITION( $G,\ M,\ \&T,\ \theta$ )
    for each region $\mathcal{R}_i \in PO$ do
        if road length in $\mathcal{R}_i > \theta$ then
            $P \leftarrow$ point closest to $\mathcal{R}_i.center$
            draw a square $Q$ of side length $l$ center at $P$
            $sl \leftarrow$ length of line segments intersecting $Q$
            j $\leftarrow$ Hash( $k, sl$ )
            if $j^{th}$ LSB of $P.x$ is 1 then
                Mark $\mathcal{R}_i$ as "match"
    STATS( $T$ )
    for each non-leaf node $T_i$ of $T$ in depth-first order do
        if $conf(T_i) > \xi$ then return $Yes$
return $No$
```

If the watermarked map is attacked by "massive crop" attack, the MQtree structure partitioned by detection algorithm is just a part of the insertion MQtree. However, it will be almost the same as the corresponding part of the MQtree for the whole map. An example of the detection process is illustrated in Figure 8. Assuming that

Algorithm 4 Compute Stats For a Given MQtree

```
procedure STATS($T$)
    if $T$ is not a leaf node then
        for each $T_i \in T.children$ do
            STATS($T_i$)
            $T$.match←$T$.match+$T_i$.match
            $T$.total←$T$.total+$T_i$.total
    else
        if The region of $T$ is "match" then
            $T$.match←1
        else
            $T$.match←0
        $T$.total←1
```

the shaded parts of Figure 8(a) is cropped from the watermarked map, the corresponding MQtree is shown in Figure 8(b).



(a) Map (b) MQtree

Figure 8: Detection Strategy

If the watermarked map is attacked by "Merge Attack", the detection MQtree will be almost the same as the insertion one. However, we can find the watermark only in part of the MQtree. In this case, the algorithm reports the sub-regions that are watermarked. We can make a depth-first traversal of the detection MQtree to find the largest watermarked region.

3.5 Analysis

The time complexity of Algorithm 1 is

$$\mathcal{T} = O(|E| \log \mathcal{L}),$$

where $\mathcal{L}$ is the total length of all roads in map M. Since the number of edges connected to a vertex is bounded a small number q, that is, $|E| \leq (q \times |V|)/2$, $\mathcal{T} = O(|V| \log \mathcal{L})$.

The time complexity of Algorithm 2 and Algorithm 3 is $\mathcal{T} + O(|V|)$, or $O(|V| \log \mathcal{L})$.

The detection confidence in (1) essentially represents the probability of correctly identifying watermarks in a given map (attacked or not). We now attempt to give a lower bound to the detection confidence. Intuitively, as θ decreases, the map is divided into more subregions and therefore more watermarks will be inserted while the accuracy of watermark detection should be enhanced. We thus have the following lemma.

LEMMA 3.1. *Given a map M with total length $\mathcal{L}$ and an algorithm threshold θ, the minimum detection confidence for M:*

$$conf_{min}(M) = 1 - \sum_{i=\lceil \rho\mathcal{L}/4\theta \rceil}^{\lceil \mathcal{L}/4\theta \rceil} \binom{\lceil \mathcal{L}/4\theta \rceil}{i} \left(\frac{1}{2}\right)^{\lceil \mathcal{L}/4\theta \rceil}$$

where ρ is the ratio between the number of leaf nodes that match and the total number of leaf nodes in M.

PROOF. Straightforward as the total number of leaf nodes is no less than $\mathcal{L}/4\theta$. □

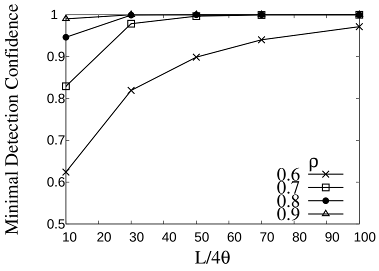


Figure 9: Accuracy of Detection

Figure 9 plots the relationship between minimum detection confidence against $\mathcal{L}/4\theta$ for different values of ρ, which coincides with our previous intuition. However, smaller θ translates into more watermarks and hence larger distortion to the map. It also causes the watermarking to be more time-consuming. This trade-off should be carefully considered when applying our approach.

4. EVALUATION

In this section, we present some experimental results from the proposed framework.

4.1 The Experimental Setup

We implement and test the performance of the algorithms under different potential attacks. All algorithms are implemented with C and all experiments are conducted on Core PC with 2.0 GHz CPU and 2 GBytes of memory running the Ubuntu 12.04 operating system.

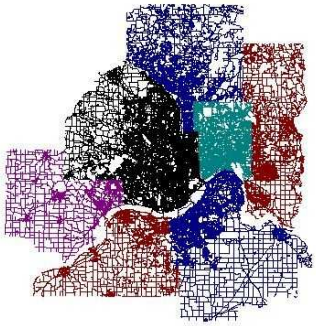


Figure 10: Twin-Cities Seven-County Road Map

We use MN state base map of seven counties, namely, Anoka, Carver, Dakota, Hennepin, Ramsey, Scott, Washington, as our real data set for this experiment. In this data-set, there are in total 415,651 line segments and 372,466 different points. Figure 10 shows the visual map of digital data for these seven counties. This data set can be downloaded from the MN/DOT web site[1].

In order to evaluate the robustness of our watermarking algorithms (which is called Jiang in the rest of this section), we illustrate the performance of our proposed watermark approach under different attacks with different settings. In this part, we compare the proposed approach with two other watermarking algorithms proposed by Pu[17] and Voigt[23]. Both methods are blind watermarking algorithms and provide some resistance to crop attack. We control the accuracy lost of the experiment data caused by different methods and compare the performance of them under noise attack, crop attack and merge attack. In this section, “crop ratio” and “merge ratio” mean the area ratio we crop from the total watermarked map. We set ξ as 0.95.

4.2 Performance under Noise Attack

Noises are added to randomly selected subsets of watermarked maps. Meanwhile, to keep the usability of a map, those subsets could not take too much proportion of the total map. In this experiment, we watermark the map of St. Paul area and attack the watermarked map with some random Gaussian noise, perturbing the map with different accuracy lost.

In this noise attack experiment, we watermark the map with algorithms of Pu[17], Voigt[23] and Jiang in certain distortion. After that we attack the watermarked maps with different noise distortion. For each noise distortion, the noise will be added in different methods for 20 times. For all three methods, the noise added is “exactly” the same each time. At last, we detect the watermark and calculate detection accuracy for three algorithms. We set distortion δ_w added by the watermarking as 10.5×10^{-6} (Jiang), 10.8×10^{-6} (Voigt) and 9.8×10^{-6} (Pu). Take the size of map into consideration, these distortion can be deemed at the same level. On the other hand, the noise distortion δ_n is changed from 5.12×10^{-3} to 2.56×10^{-2}. Actually, δ_n here is large enough that almost changes all vertexes of the map. Figure 11 shows the results.

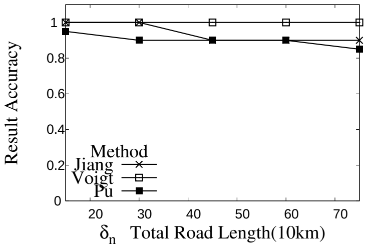


Figure 11: Performance under Noise Attack

In this experiment, we set the standard of positive detection as correctly decide the whole map is watermarked. However, in fact, according to the detection method of our algorithm, some more assistance can be obtained. Even the whole map is failed to be detected, the result could be a series of sub-regions that is suspicious. The results show that our algorithm successfully survived noise attack. On one hand, we watermark certain bits which will not be directly affected by the noise if perturbing for individual vertex is under certain strength. Meanwhile, even the perturbing for individual vertex is powerful enough, θ and the hash function of sl in in partition and insertion algorithms provide a good tolerance for it.

4.3 Performance under Crop Attack

In our experiment, on one hand, each county out of seven is selected to be detected. On the other hand, some certain proportion of the map is selected to evaluate the performance of our algorithm.

In this experiment, the parameter θ is set as 30,000.0 ($\delta_w = 10.5 \times 10^{-6}$) and the square side length is 100.0 for both insertion and detection. We partition the whole map of Minneapolis-St. Paul Metropolitan area. Then the watermarked map is split according to the boundary of different counties. After that, we impose our watermarking algorithm on these “partial” maps, trying to decide whether the maps are parts of our original map. Table 2 shows the results of this experiment.

Table 2: Crop Attack Detection

County	Total	Match	Confidence	Result
Anoka	55	51	1.000	positive
Carver	28	23	0.999	positive
Dakota	68	61	1.000	positive
Hennipin	141	137	1.000	positive
Ramsey	58	56	1.000	positive
Scott	29	25	0.999	positive
Washington	46	43	1.000	positive

By analyzing the insertion algorithm, we can easily get the notion that the partition criterion θ has a large influence on the result of our watermark algorithm. We also design a series of experiments to figure out the influence of the θ. In this experiment, we watermark the map with different distortion (2.1) by adjusting θ (see Figure 13(a)) and then select the crop ratio of $1/2$, $1/4$ and $1/8$ of the map to detect. Each ratio is detected for 20 times. Figure 13(b) show the result of experiments. Insertion (Figure 13(c)) and detection (Figure 13(d)) time for different distortion is also measured.

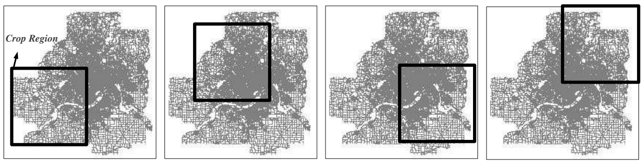
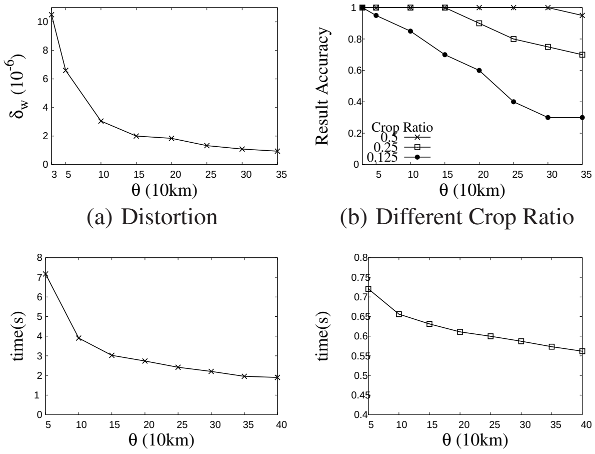


Figure 12: A Showcase of Crop Ratio 1/4



(a) Distortion

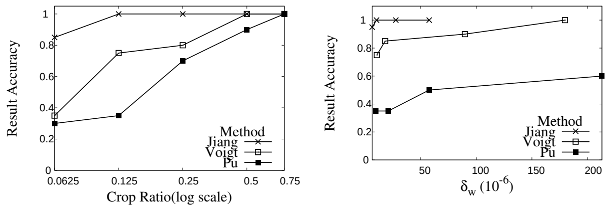
(b) Different Crop Ratio

(c) Time of Insertion

(d) Time of Detection

Figure 13: Experiment Results under Different Distortion

In the following experiment, we watermark a map with certain distortion, here we set $\theta = 30,000$. After that we crop different ratio from the map to evaluate the performance of our algorithm. For each crop ratio, we select different parts of the map to detect up to 20 times. We randomly select different parts of the watermarked map to detect (see Figure 12) and calculate the detection accuracy. Here we also implement Voigt’s and Pu’s methods to make a comparison. Distortion δ_w added by the watermarking is 10.5×10^{-6} (Jiang), 10.8×10^{-6} (Voigt) and 9.8×10^{-6} (Pu). The distortion of all three methods are almost at the same level. Figure 14(a) gives the results.



(a) Performance under Cropping (b) Accuracy in $1/8$ crop ratio

Figure 14: Performance under Crop Attack

According to the experiment result, we can find that our algorithm keeps a perfect result when the crop ratio is up to $1/8$, while the comparison methods show a significant accuracy degradation. And when crop ratio is $1/16$, we still contain a high accuracy. We also test the detection accuracy under different distortion in $1/8$ crop ratio (see Figure 14(b)). In this experiment, our algorithm can attain a 100% detection accuracy with just little distortion. Since the detection accuracy of our algorithm quickly attain to 1, it is not necessary to plot too many points in Figure 14(b).

4.4 Performance under Merge Attack

In this experiment, we select the 2012 TIGER map of Minneapolis-St. Paul Metropolitan area as another data source, which can be downloaded from the United States Census Website[2]. The coordinate system of TIGER map is different from MN dot base Map. We transform the coordinate system of TIGER map to make it same as the former data source. Figure 15 is the overlap of these two maps of Anoka county after the synchronization.

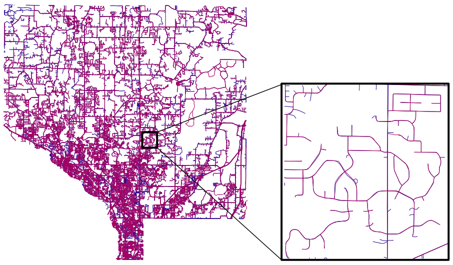


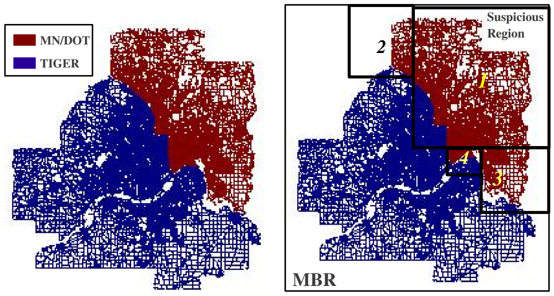
Figure 15: Overlap of Two Maps

We crop a watermarked map ($\delta_w = 6.60 \times 10^{-6}$) and merge part of it with TIGER map to make a "new" map (see Figure 16(a)). The result of the merge detection: suspicious regions and detection confidence is shown in Table 3. Here we select those regions which have at least 10 points tested. From the corresponding visual figure (see Figure 16(b)), we can see that our algorithm almost "exactly" decides the suspicious regions.

Table 3: Merge Attack Detection

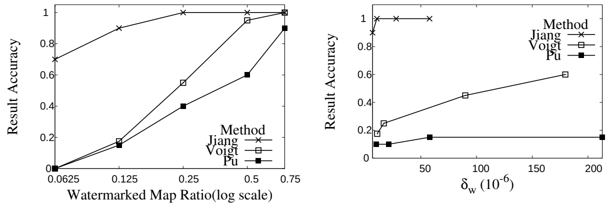
Region	Total	Match	Confidence	Result
1	147	122	1.000	positive
2	12	11	0.997	positive
3	23	18	0.994	positive
4	15	12	0.983	positive

We also watermark the Minneapolis-St. Paul Metropolitan area and then split this watermarked map into the same small portions as cropping experiment above. After that we merge these maps with the complementary parts of map from TIGER to create some new digital road maps. In this experiment, we also implement the two other method to make a comparison. Distortion δ_w added by the watermarking still is 10.5×10^{-6} (Jiang), 10.8×10^{-6} (Voigt) and 9.8×10^{-6} (Pu). Figure 17(a) shows the result of our experiments.



(a) Merge Attack (b) Detection Result

Figure 16: Merged Map



(a) Performance under merging (b) Accuracy in $1/8$ merge ratio

Figure 17: Performance under Crop Attack

Comparing with the cropping experiment, we can obviously find the performance of our proposed algorithm is almost not affected by the complementary part of the map. However, the detection accuracy of the other two methods decrease significantly. The reason is the complementary map provides sufficient disturb to watermarked data. Similarly, we also implement another experiment to evaluate the relationship between detection accuracy and watermark distortion δ_w. From this figure, we can see that our algorithm still can quickly attain a quite high detection accuracy. However, when δ_w increases for Pu's and Voigt's method, the detection accuracy doesn't changed significantly. The reason for Pu should be that it's global liner correlation is destroyed by "merge" attack. For Voigt's method, though more watermark information is used for detection, more "merge" noise will will also be added to the detection process. Thus increase of detection accuracy for them is not significant.

5. RELATED WORK

Many efforts have been made to protect the copyrights of multimedia products such as images, movies and music. Digital road maps is a special vector graphics data which contains important geographical and spatial information. In the rest of this section, we will focus on GIS spatial data watermarking but will also briefly touch on related techniques in some other domains. The technique proposed in this paper benefits from or is inspired by some of these methods.

5.1 GIS Spatial Data Watermarking

Several studies have touched on the watermarking of GIS spatial data such as digital vector maps. Even though digital vector maps are represented by node coordinates which can also be treated as relational tables, techniques developed for relational data ignore the spatial properties of road maps such as spatial distances and geo-localities. These distinct features of GIS spatial data call for special treatments in watermarking techniques. According to Niu [14], the existing methods of digital vector map watermarking is classified into two categories: algorithms in spatial domain and algorithms in frequency domain. Depending on whether to require the original map for detection, these methods are also classified as the not blind and the blind. In addition, algorithms can also be categorized into *global* or *local* depending on whether global information of the map is needed when computing each watermark.

The spatial domain algorithms embed watermarks based on the geometric properties of polyline and polygon objects. It is often easier to control the amount of distortion added by the watermarks. They are also more robust against rotation, scaling and noises, while preserving the utility of a map. Existing methods in spatial domain usually lack of robustness against massive cropping and merging. The method proposed in this paper falls into the category of spatial domain algorithms.

The frequency domain algorithms essentially transform the original data into frequency domain, usually by Discrete Fourier Transform (DFT) [11, 21, 22] or Discrete Wavelet Transform (DWT) [13], add noises in the frequency domain and then transform it back into the spatial domain. The critical drawbacks of frequency domain techniques are the difficulty of controlling the amount of distortion in the spatial domain and the lack of robustness against data cropping and data reordering, due to the fact that it is actually a global watermarking scheme.

5.1.1 Non-Blind Methods in Spatial Domain

Some watermarking methods [7, 15, 16] require the original map or watermarked map as a reference to detect watermarks from a suspicious map. This kind of methods are called "non-blind" algorithms. The main weakness of these methods is that road maps can be updated over time and a map producer may have many versions of the same map in its database. Moreover in the same database, there could be many maps which have overlapping regions, say map of the Twin Cities, map of Minnesota, map of the United States, etc. When the map database is big, and when the suspicious map could have been subjected to cropping and merging, it is not easy to identify the original map.

Ohbuchi et al.[15] partitions the space of the digital map into rectangles such that every rectangle contains almost equal node numbers. Then the v^{th} vertex inside the rectangle is modified to include a watermark. In another method [16], all vertices are connected into a single mesh by Delaunay triangulation. Then a mesh Laplacian is formed and the mesh is partitioned into patches using the same method as in the former one[15]. Mesh-spectral coefficients are calculated for every patch and the watermark is embedded into these coefficients.

Both of the above algorithms disperse watermarks locally and have some robustness against crop attack. However, them are highly dependent on the validity of the original data. When extracting the watermark, the suspicious map is aligned onto the original map. All inserted nodes have to be deleted, and all removed nodes have to be correctly recovered.

5.1.2 Blind Methods in Spatial Domain

Some other watermark framework are proposed which required no original or watermarked map as a reference. Some of these methods [8, 9, 24, 12] insert watermark globally. Yan et al. proposed a key point based algorithm [24]. Key points are those nodes in the map which have more important geometric aspects than the other ones, for example cross joints and those in sharp curve. They are not likely to be removed or edited by the attackers because removing them will render the map useless. This method uses a dynamic programming algorithm [6] both in insertion and detection steps. Key points in the map are used for inserting and detecting watermarks. This kind of algorithms are robust against some attacks like noise and vertex simplification, but are still vulnerable to crop and merge attacks.

Other blind watermarking algorithms [23, 18, 17, 5, 10] provide some limited resistance against crop attacks to some extent. Voigt et al.[23] proposed a feature based watermarking algorithm which is relies on statistical detection. This method partitions a map into small rectangular regions called "patches". It then randomly selects two subsets of the patches called set A and set B, respectively. Next, it calculates a reference point for each patch in the two set. During watermark insertion, all nodes in set A are shifted *toward* the reference point in their respective patches, while all nodes in set B are shifted *away* from the reference point. The amount of shift is governed by an F-distribution [3] which is also used to detect the watermark. Pu et al. proposed a blind algorithm [17] which divides the map into mesh segments, and then embeds the watermark in each segment with a fixed order. In the detection step, each segment is evaluated by a correlation parameter, which is a linear correlation of the watermark and the watermarked data. To survive crop attack, a global correlation based detection method is applied as an optimization. This algorithm watermarks maps according to their topological relations.

While these algorithms present some resistance against cropping attack, they cannot survive cropping at larger scale. Take Voigt[23] as an example, when the watermarked map is massively cropped, many patches that are marked may be removed. Thus F distribution of watermarked points may be destroyed. Therefore, the method is not robust enough against "massive crop". Furthermore, if the map is cropped and then merged with others, the information added by the merge will defeat the watermarking more easily.

5.1.3 Methods in Frequency Domain

Solachidis et al.[21] proposed a blind watermarking scheme embedding a single bit into a polyline by modifying the discrete Fourier coefficients of polyline's coordinate sequence. This method embeds the watermark in the magnitude of the curve's Fourier descriptors to exploit its location, scale, and rotation invariant properties. Due to the amplitude frequency features of discrete Fourier transform, the algorithm is inherently robust to many attacks such as map translation, rotation, scaling and changing start vertex. Similar to the DFT method[21], Li et al. proposed a blind scheme [13] embedding multi-bits into a vector map in DWT domain. These frequency domain methods rely on the integrity of map and order of data points. When part of map data is removed or the order of data points is changed, the coefficients will also be changed, which makes them extremely vulnerable to crop and merge attacks.

5.2 Watermarking in Other Domains

Some research attention has been given to relational data[4, 20]. Rakesh Agrawal proposes a framework of watermarking relational databases[4]. According to this approach, a primary key is stored in some significant tuples of relational data. The altered attribute index and bit index for the selected attribute are randomly selected according to secret keys. Their approach is highly dependent on the primary key of the data tuple and randomly selects a subset of tuples to watermark. This approach in fact disperses one bit watermark into different position of relational data. It provide some resistance against crop and merge attack to some extent. Some ideas in this framework actually inspired the work in this paper, e.g., modifying least significant bits and computing the confidence of detection. However, this framework is for relational data and hence does not consider the special properties of spatial attributes in GIS spatial data. In our method, information which is adjacent to each other in geographical space collaborate to decide the watermarking positions. However, in relational data, different tuples are completely independent to each other, even though they are stored together, because they form a set. Other work[20] extends the framework[19] to relational databases. In that paper, numeric dataset are first hashed to another dataset with a secret order. Then the new dataset is grouped into different chunks according to their order. Finally, the average value and standard deviation is calculated for every chunk. According to these statistics and watermarking data, a small number of data points are altered. This approach still disperses the watermark globally. One of the serious drawbacks for this global watermarking approach is that it completely fails a "massive crop" attack.

6. CONCLUSION

In this paper, we proposed a new blind watermarking scheme for digital vector road maps. The scheme produces and detects watermarks according to local information with the help of three secret keys of negligible sizes but without referring to the original map. The algorithm dynamically partitions a given map according to road density and inserts one-bit watermarks to one of the least significant bits of points determined by the secret keys. The amount of distortion brought by watermarks is arguably much smaller than existing methods. Our preliminary evaluation shows that this algorithm is resilient to massive crop and merge attacks and significantly outperforms two other state-of-the-art vector map watermarking approaches in terms of detection accuracy.

7. REFERENCES

[1] MN/DOT. `http://www.dot.state.mn.us/`.

[2] TIGER. `http://www.census.gov/`.

[3] M. Abramowitz and I. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Dover publications, 1965.

[4] R. Agrawal and J. Kiernan. Watermarking relational databases. In *VLDB*, pages 155–166, 2002.

[5] S. Bird, C. Bellman, and R. v. Schyndel. A shape-based vector watermark for digital mapping. In *Proceedings of the 2009 Digital Image Computing: Techniques and Applications*, DICTA '09, pages 454–461, Washington, DC, USA, 2009. IEEE Computer Society.

[6] D. H. Douglas. Algorithms for the reduction of the number of points required to represent a line or its a caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[7] P. Han, J. Gong, and L. Cheng. An improved adaptive watermarking algorithm for vector digital maps. In *Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on*, pages 2844 –2847, 31 2006-aug. 4 2006.

[8] X.-J. Huo, T.-Y. Seung, B.-J. Jang, S.-H. Lee, and K.-R. Kwon. A watermarking scheme using polyline and polygon characteristic of shapefile. In *Intelligent Networks and Intelligent Systems (ICINIS), 2010 3rd International Conference on*, pages 649 –652, nov. 2010.

[9] H. Kang, K. Kim, and J. Choi. A vector watermarking using the generalized square mask. In *ITCC*, pages 234–236. IEEE Computer Society, 2001.

[10] J. Kim, S. Won, W. Zeng, and S. Park. Copyright protection of vector map using digital watermarking in the spatial domain. In *Digital Content, Multimedia Technology and its Applications (IDCTA), 2011 7th International Conference on*, pages 154 –159, aug. 2011.

[11] I. Kitamura, S. Kanai, and T. Kishinami. Copyright protection of vector map using digital watermarking method based on discrete fourier transform. In *Geoscience and Remote Sensing Symposium, 2001. IGARSS '01. IEEE 2001 International*, volume 3, pages 1191 –1193 vol.3, 2001.

[12] S.-H. Lee and K.-R. Kwon. Vector watermarking scheme for GIS vector map management. *DOI10.1007/s11042-011-0894-y SpringerLink*, 2011.

[13] Y. Li and L. Xu. A blind watermarking of vector graphics images. In *ICCIMA '03: Proceedings of the 5th International Conference on Computational Intelligence and Multimedia Applications*, page 424, Washington, DC, USA, 2003. IEEE Computer Society.

[14] X. Niu, C. Shao, and X. Wang. A survey of digital vector map watermarking. In *2006 International Journal of Innovative Computing,Information and Control Volume 2*, 2006.

[15] R. Ohbuchi, H. Ueda, and S. Endoh. Robust watermarking of vector digital maps. In *ICME (1)*, pages 577–580, 2002.

[16] R. Ohbuchi, H. Ueda, and S. Endoh. Watermarking 2d vector maps in the mesh-spectral domain. In *Shape Modeling International*, pages 216–228, 2003.

[17] Y.-C. Pu, W.-C. Du, and I.-C. Jou. Toward blind robust watermarking of vector maps. In *ICPR (3)*, pages 930–933, 2006.

[18] G. Schulz and M. Voigt. A high capacity watermarking system for digital maps. In *MM&Sec*, pages 180–186, 2004.

[19] R. Sion, M. J. Atallah, and S. Prabhakar. On watermarking numeric sets. In *IWDW*, pages 130–146, 2002.

[20] R. Sion, M. J. Atallah, and S. Prabhakar. Rights protection for relational data. In *SIGMOD Conference*, pages 98–109, 2003.

[21] V. Solachidis and I. Pitas. Watermarking polygonal lines using fourier descriptors. *Computer Graphics and Applications, IEEE*, 24(3):44 – 51, may-jun 2004.

[22] H. Sonnet, T. Isenberg, J. Dittmann, and T. Strothotte. Illustration watermarks for vector graphics. *Computer Graphics and Applications, Pacific Conference on*, 0:73, 2003.

[23] M. Voigt and C. Busch. Feature-based watermarking of 2d-vector data. In *SPIE,Security and Watermarking of Multimedia Content.Santa Clara,*, pages 359–366, 2003.

[24] H. Yan, J. Li, and H. Wen. A key points-based blind watermarking approach for vector geo-spatial data. *Computers, Environment and Urban Systems*, 35(6):485–492, 2011.