International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.4 (2015), pp.155-172
http://dx.doi.org/10.14257/ijmue.2015.10.4.16

RST Invariant Fragile Watermarking for 2D Vector Map Authentication

Nana Wang[1], Jilong Bian[2] and Han Zhang[3]

Harbin Engineering University[1]
Northeast Forestry University[2]
Heze University
wangnana_5@yahoo.com[1], bianjilong@hrbeu.edu.cn[2], zhanghan_6@yahoo.com

Abstract

2D vector maps are very important data in geographical information systems (GISs). It is considered to be content for which verification of integrity and authentication are urgently required. Using a marking technique and a rotation, scaling and translation (RST) invariant watermark generation method, we propose a scheme that detects and locates malicious attacks with high accuracy while providing the RST invariance property. In particular, we propose setting a unique mark on each feature, calculating the authentication watermark for each spatial feature group by folding the hash results of the differences of the log-radiuses, and embedding the watermark using a RST invariant watermarking method. While the watermark ensures both the sensitivity to malicious manipulations and the RST invariance property, the mark of each feature provides superior accuracy of tamper localization. Moreover, this algorithm is immune to feature rearrangement and vertex reversing operations. Theoretical analysis and experimental results are provided to demonstrate the effectiveness of our method.

Keywords: *authentication, fragile watermarking, tamper localization, RST invariance property, 2D vector map*

1. Introduction

These days, digital products are gradually replacing their classical analogue counterparts. This is quite understandable because digital data can be easily replicated, manipulated and distributed using powerful available tools and equipment. On the other hand, it is very easy even for an amateur to illegally produce perfect copies of the original data and maliciously modify the original content. Thus, it is desirable to develop a strong method to protect the copyright and authenticate the integrity of the digital data.

Many view watermarking as a potential solution for the requirement above. This technique can be defined as a process that embeds a secret code, called a watermark, into an object. Depending on the end applications, watermarking can be classified as robust or fragile schemes. The robust watermarking indicates the ownership while the fragile watermarking authenticates the integrity of digital data and locates any modification made to the original content. In this paper, we mainly consider fragile watermarking schemes for 2D vector maps.

Currently, research on watermarking for 2D vector maps is mainly focused on robust watermarking [1-9] for copyright protection or reversible (also referred to as invertible, lossless, or distortion-free) watermarking [10-14] for content recovery. Few works have been done on 2D vector map fragile watermarking [15-19]. Shao *et al.* [15] divided the vertices into non-intersecting groups and embedded the watermark using Fridrich *et al.'s* reversible watermarking algorithm [14]. A drawback of this algorithm is that the tamper localization ability will be ruined if some vertices/features have been added or deleted.

ISSN: 1975-0080 IJMUE

Zheng and You [16] separated the vertices of a 2D vector map into several blocks according to a predefined threshold and embedded the watermark block by block. Because a vertex addition/deletion attack may make the block division result different from the original one, leading to correct watermark extraction impossible, it fails to locate tampered blocks. Another scheme was presented by Zheng *et al.* [17], in which the features are divided into different groups and the watermarks are embedded group by group independently. As the features are traversed according the record number, a feature reordering operation or a feature addition/deletion attack may totally disable the localization capability. To locate malicious attacks accurately and recover the original content, Wang and Men [18] proposed a reversible fragile watermarking algorithm based on feature marking. The features are divided into groups and the watermark is embedded using Wang et al.'s reversible watermarking technique [10]. For achieving good tamper localization performance, a marking method based on vertex insertion is applied to each feature. This scheme can locate tampered data accurately. Besides, Wang and Men [19] proposed a reversible fragile watermarking scheme for locating tampered blocks using a fragile watermarking technique based on vertex insertion. It can locate tampered regions accurately. However, since in some applications a vector map is still regarded as having value in use after RST transformations, and these operations are not seen as malicious attacks, the lack of RST invariance property may limit these schemes' application scope.

To provide both the RST invariance property and good tamper localization capability, we propose a RST invariant authentication watermark generation method and a feature location marking technique, and apply them to the fragile watermarking scheme for 2D vector maps. The shapefile format of Environmental Systems Research Institute, Inc. (ESRI) [20] is exploited for the algorithm. We divide the spatial features into groups and apply the marking method to each feature for correctly identifying the original features of each group in the watermark verification stage. After that, for each group, we generate a RST invariant watermark by folding the hash results of the differences of the log-radiuses, and embed the watermark using Chou and Tseng's method [21]. While the watermark generation method ensures the properties of RST invariance and the sensitivity to malicious manipulations, the feature marking algorithm provides good tamper localization capability. Moreover, this scheme is immune to feature rearrangement and vertex reversing operations.

The remaining sections are organized as follows. Section 2 briefly reviews the watermarking method by Chou and Tseng [21]. Section 3 explains our RST invariant fragile watermarking algorithm in detail. We present the experimental results and an analysis of the algorithm in Section 4. Conclusions are summarized in Section 5.

2. Chou and Tseng's RST Invariant Watermarking Method

The 3D triangle model is used as the cover in Chou and Tseng's algorithm [21]. Let t be a triangle, v_n, v_w and v_h be the three vertices of t, w ($0 \le w < S_w$, $S_w = 1, 2, 3,\ldots$) be the watermark to be embedded, S_w be an embedding parameter.

In the embedding phase, the functionalities of t's three vertices (v_n, v_w and v_h) are first assigned. The vertex v_n, called the *normalization* vertex, is used to get the normalized quantization step for t. The vertex v_w, called the *watermark-embedding* vertex, is used to embed the watermark w. The vertex v_h, called the *hash-embedding* vertex, is used to embed the hash function value $h(w)$. w and $h(w)$ are embedded and extracted using the same method.

For embedding the watermark w into the vertex v_w, the position of t's neighboring vertices' center t^c is first calculated,

$$t^c = \frac{1}{\left|N(v)\right|} \sum_{v_i \in N(v)} v_i \,, \tag{1}$$

where $N(v)$ is the set of t's neighboring vertices and $|N(v)|$ is the size of $N(v)$. In the following, we call the center of t's neighboring vertices, *i.e.*, t^c, the *neighboring center*.

Then, the normalized watermark quantization step Q_w is computed using the Euclidean distance $||v_n t^c||$ between the *normalization* vertex v_n and the *neighboring center* t^c,

$$Q_w = \left\| v_n t^c \right\| / K_w, \tag{2}$$

where K_w is an embedding parameter used to control the maximum distortion.

After that, according to the Euclidean distance $||v_w t^c||$ between the *watermark-embedding* vertex v_w and the *neighboring center* t^c, v_w is moved to a new location v_w^e so that the Euclidean distance $|| v_w^e t^c||$ between v_w^e and t^c is a multiple of Q_w,

$$v_w^e = v_w - \frac{v_w - t^c}{\left\| v_w t^c \right\|} \cdot (\left\| v_w t^c \right\| \mod Q_w). \tag{3}$$

Finally, the watermark w is embedded into v_w^e and the watermarked vertex v_w' is obtained,

$$v_w' = v_w^e + \frac{v_w^e - t^c}{\left\| v_w^e t^c \right\|} \cdot \frac{Q_w}{S_w} \cdot w. \tag{4}$$

After watermark embedding, v_w' is on the straight line that is determined by v_w and t^c. For embedding the watermark w, v_w moves less than Q_w, *i.e.*, the Euclidean distance $||v_w v_w'||$ between v_w and v_w' is less than Q_w ($||v_w v_w'|| < Q_w$). Thus, the embedding distortion is less than Q_w.

The above is the embedding procedure.

In the extraction phase, the normalized watermark quantization step Q_w' is first calculated using Eq. (2).

Then, according to the Euclidean distance $||v_w' t^c||$ between v_w' and t^c, the watermark is extracted from v_w' by the following equation:

$$w = (\left\| v_w' t^c \right\| \mod Q_w') \cdot \frac{S_w}{Q_w'}. \tag{5}$$

The above is Chou and Tseng's method. After applying the RST transformations to a watermarked model, the embedded watermark can still be correctly extracted.

3. The Proposed Watermarking Scheme

In the proposed scheme, we suppose that the 2D vector map is composed of polyline or polygon features. As the only difference between a polyline and a polygon is that the first vertex coincides with the last one in a polygon, we will describe the method for polylines in detail.

The scheme is introduced in two stages: watermark embedding and watermark verification.

In the watermark embedding stage, the polylines of the vector map are first divided into groups using the watermark length L and the number of watermark bits a vertex carries c. As the original polylines of each group may be difficult to be

identified after malicious attacks, to accurately locate tampered groups in the watermark verification stage, we assign a unique mark, called the *location ID*, to each polyline and embed it into the polyline in the second step. The *location ID* can not only indicate the group which the polyline is divided into, but also denote the polyline's location in the group. The vertices used to indicate the *location ID* are called *mark* vertices. Because of the importance of identifying the polylines' original locations in the verification phase, the *mark* vertices do not carry watermark bits. Then, we generate a RST invariant authentication watermark for each group under the control of two private keys: *K* and *k*, and embed the watermark using Chou and Tseng's method [21]. In each group, only one polyline is used to carry the watermark. Since a vertex carries *c* bits and the number of the *mark* vertices of each polyline is 4 which will be described in the feature location marking stage, the whole number of vertices within the polyline, which is used to carry the watermark, should be at least $\lceil L/c \rceil + 4$. We call the polyline that contains at least $\lceil L/c \rceil + 4$ vertices an *eligible* polyline.

In the watermark verification stage, we first extract the embedded *location ID* of each polyline to identify its original location using the total number of polylines *Z*(*Z* = 1, 2, 3,…) and the parameter K_w. Then, according to the polylines' *location ID*s, the original polylines of each group can be obtained. After that, for each group, we extract the embedded watermark and derive a watermark from the received vector map. Finally, tampered groups can be located by judging the mismatch between the extracted watermark and the watermark derived from the received content.

3.1 Watermark Embedding Procedure

The watermark embedding process consists of four basic steps: i) Group division ii) Feature location marking iii) Authentication watermark generation and iv) Watermark embedding.

3.1.1 Group Division: Given a 2D vector map *M* that has *Z* polylines, we first traverse the whole vector map to find *eligible* polylines according to the unique record numbers of the polylines. Denote the polyline list which is ordered according to the polylines' unique record numbers as $P = \{P^i \mid i \in [0, Z-1]\}$, the *eligible* polyline list which is ordered according to their unique record numbers as $P_e = \{P_e^{\,i} \mid i \in [0, N_e - 1]\}$, and the whole number of *eligible* polylines of the vector map *M* as N_e.

Then, we divide the polylines of *P* into non-overlapped groups, each of which contains *n* ($n \geq 1$) polylines and at least one *eligible* polyline. To make sure that each group contains at least one *eligible* polyline, we assume N_e is greater than or equal to the whole number of groups N_g ($N_g = \lceil Z/n \rceil$). Because *Z* may be not a multiple of *n*, the number of polylines in the last group may be less than *n*. Suppose $P_e^{\,s} = \{ P_e^{\,s,i} \mid i \in [0, N_g - 1]\}$ is the *eligible* polyline list that contains the first N_g ordered polylines of P_e, and $P_{ne} = P - P_e^{\,s} = \{P_{ne}^{\,i} \mid i \in [0, Z - N_g - 1]\}$ is the polyline list which consists of the ordered polylines of *P* except the polylines of $P_e^{\,s}$. We divide the polylines into non-overlapped groups using the following method.

For the *i*-th ($0 \leq i < N_g - 1$) group G_i, it consists of the *i*-th *eligible* polyline of $P_e^{\,s}$ (*i.e.*, $P_e^{\,s,i}$) and P_{ne}'s *n* –1 polylines from $P_{ne}^{\,i\times(n-1)}$ to $P_{ne}^{\,(i+1)\times(n-1)-1}$.

For the last group G_{Ng-1}, it consists of the (N_g–1)-th *eligible* polyline of $P_e^{\,s}$ (*i.e.*, $P_e^{\,s,\,Ng-1}$) and P_{ne}'s N_t (N_t =Z–N_g–(N_g–1)×(*n*–1)) polylines from $P_n^{\,(Ng-1)\times(n-1)}$ to $P_n^{\,Z-Ng-1}$. That is, after group division, we have

$$\begin{cases} G_i = \{P_e^{s,i},\ P_{ne}^{j} \mid j \in [i\times(n-1),\ (i+1)\times(n-1)-1]\}, & 0 \leq i < N_g - 1 \\ G_i = \{P_e^{s,i},\ P_{ne}^{j} \mid j \in [i\times(n-1),\ Z - N_g - 1]\}, & i = N_g - 1 \end{cases}. \quad (6)$$

We call the first *eligible* polyline of each group, which is used to carry the watermark, a *watermark* polyline, the second vertex of the *watermark* polyline a *reference*1 vertex and the penultimate vertex of the *watermark* polyline a *reference*2 vertex. The *reference*1 vertex and the *reference*2 vertex will be used in the watermark embedding step.

3.1.2 Feature Location Marking: For each group, a unique mark, *i.e.*, the *location ID*, is assigned to each polyline. As a simple implementation, we assign the *location ID* $m_{i,j}$ ($m_{i,j} = i \times n + j$, $m_{i,j} \in [1, Z]$) to the j-th ($1 \le j \le n$) polyline of the i-th ($0 \le i \le N_g - 1$) group G_i.

Then, we divide the polylines into three categories: one is composed of more than three vertices (*normal*), one is composed of three vertices (*complex*1) and the other is composed of two vertices (*complex*2). We will mark the three types of polylines using different approaches.

a. Method of marking the location for a *normal* polyline

Let's assume a *normal* polyline $Pl_n = \{v_i | \ i \in [0, r-1], r > 3\}$ contains r vertices and its *location ID* is $m_{p,q}$ ($1 \le q \le n$, $0 \le p \le N_g - 1$, $m_{p,q} \in [1, Z]$). We embed $m_{p,q}$ into v_0 to mark its *location ID*, and embed $m_{p,q}+1$ into v_{r-1} to indicate the vertex order.

We use Chou and Tseng's method [21] to embed $m_{p,q}$ and $m_{p,q}+1$. First, $m_{p,q}$ is embedded into v_0. According to Eqs. (2)-(4), three different vertices (a *normalization* vertex, a *watermark-embedding* vertex and a *neighboring center*) are needed to embed $m_{p,q}$. As $m_{p,q}+1$ is going to be embedded into v_{r-1}, v_{r-1}'s coordinates will be changed after embedding. If v_{r-1} is used as a *normalization* vertex or a *neighboring center* in the process of embedding $m_{p,q}$, the embedded mark $m_{p,q}$ may not be extracted correctly. Here, we embed the mark $m_{p,q}$ into v_0 by regarding the vertices v_{r-2}, v_0 and v_1 ($v_0 \neq v_1 \neq v_{r-2} \neq v_{r-1}$) as the *normalization* vertex, the *watermark-embedding* vertex and the *neighboring center*, respectively.

During embedding, the parameter S_w is set as

$$S_w = Z + 2, \tag{7}$$

and the parameter K_w is set as

$$\begin{cases} \max_{len} = \max(len_{m_{i,j}}), \ (0 \le i \le N_g - 1, \ 1 \le j \le n, \ 1 \le m_{i,j} \le Z) \\ K_w = \dfrac{\max_{len}}{\tau} \end{cases}, \tag{8}$$

where $len_{m_{i,j}}$ represents the length of the polyline whose *location ID* is $m_{i,j}$, max_{len} is the maximum length of the polylines of the vector map M, and τ is the vector map's precision tolerance. With this configuration, the Euclidean distance $\|v_0 v_0'\|$ between v_0 and v_0' (the embedded v_0) is always less than τ, *i.e.*,

$$\|v_0 v_0'\| < Q_{w,1} = \frac{\|v_{r-2} v_1\|}{K_w} = \frac{\|v_{r-2} v_1\|}{\max_{len}} \times \tau < \tau. \tag{9}$$

As the distortion introduced by feature location marking does not exceed the precision tolerance τ, the validity of the marked map data can be ensured. The parameter K_w obtained here will be directly used as an input in the verification phase.

Then, according to Eqs. (2)-(4) and Eqs. (7)-(8), we embed $m_{p,q}$+1 into v_{r-1} by regarding the vertices v_1, v_{r-1} and v_{r-2} ($v_{r-2} \neq v_{r-1} \neq v_1 \neq v_0$) as the *normalization* vertex, the *watermark-embedding* vertex and the *neighboring center*, respectively.

After setting a *location ID* on a *normal* polyline Pl_n, the marked *normal* polyline Pl_n^m has the following characteristics: i) the vertex at one end of the polyline carries the polyline's unique mark $m_{p,q}$ while the vertex at the other end of the polyline carries $m_{p,q}$+1, ii) the vertex carrying $m_{p,q}$ is the first vertex of the polyline and the vertex carrying $m_{p,q}$+1 is the last vertex of the polyline and iii) if the marked *normal* polyline is composed of four vertices, neither the first three adjacent vertices nor the last three adjacent vertices of the polyline are on the same straight line.

Because of the condition, *i.e.*, $v_0 \neq v_1 \neq v_{r-2} \neq v_{r-1}$, used in the embedding of $m_{p,q}$, this marking method requires that the polyline contains at least 4 vertices. Thus, it can neither be applied to a *complex*1 polyline nor a *complex*2 polyline, and designing marking methods for *complex*1 polylines and *complex*2 polylines is necessary.

As the first two adjacent vertices and the last two adjacent vertices of a marked *normal* polyline are used to indicate the polyline's *location ID* and the vertex order, the number of its *mark* vertices is 4.

b. Method of marking the location for a *complex*1 polyline

For a *complex*1 polyline, we mark its *location ID* using Wang and Men's method [18] for marking a PL1. Assuming the location *ID* of a *complex*1 polyline Pl_{c1}= $\{v_i (v_i^x, v_i^y) | i \in [0, 2]\}$ is $m_{p,q}$, we embed $m_{p,q}$ by inserting a vertex $v_{c1}(v_{c1}^x, v_{c1}^y)$ between $v_0 (v_0^x, v_0^y)$ and $v_1(v_1^x, v_1^y)$.

The position of the vertex $v_{c1}$is calculated by

$$\begin{cases} \dfrac{v_0^x - v_{c1}^x}{v_0^x - v_1^x} = \dfrac{m_{p,q}}{Z+1} \\ \dfrac{v_0^y - v_{c1}^y}{v_0^y - v_1^y} = \dfrac{m_{p,q}}{Z+1} \end{cases}. \qquad (10)$$

In other words, the Euclidean distance between v_0 and v_1 is divided into Z+1 equal intervals and the number of intervals between v_0 and v_{c1} is used to indicate the polyline's *location ID* $m_{p,q}$.

After inserting the vertex v_{c1} into the polyline Pl_{c1}, the marked *complex*1 polyline Pl_{c1}^m has the following characteristics: i) the four vertices it contains are not on the same straight line and ii) three adjacent vertices at one end of the polyline are on the same straight line, and the end which the three adjacent vertices are at indicates the beginning of the polyline's vertex order.

Since all the four vertices of a marked *complex*1 polyline are used to indicate the polyline's *location ID* and the vertex order, the number of its *mark* vertices is 4.

Although this marking method can also be used to mark the location of a *normal* polyline, because of the extra vertex's space it requires for each polyline, it is not suitable for the applications where the storage space or the bandwidth is limited. Thus, the marking method for *normal* polylines is necessary.

Moreover, this marking method cannot be applied a *complex*2 polyline. Assuming we mark a *complex*2 polyline's *location ID* using this method, the marked *complex*2 polyline will consist of three vertices which are in the same line. After a vertex reversing attack, the original vertex order cannot be identified and the *location ID* we calculated in the verification phase may be incorrect. Thus, a marking method for *complex*2 polylines is needed.

c. Method of marking the location for a *complex*2 polyline

For a *complex*2 polyline, we mark its *location ID* using Wang and Men's method [18] method for marking a PL2. Assuming the *location ID* of a *complex*2 polyline Pl_{c2}= $\{v_i(v_i^x, v_i^y) | i \in [0, 1]\}$ is $m_{p,q}$, we insert two vertices $v_{c2,1}$ ($v_{c2,1}^x$, $v_{c2,1}^y$) and $v_{c2,2}$($v_{c2,2}^x$, $v_{c2,2}^y$) between v_0 and v_1. The vertex $v_{c2,1}$ is used to indicate Pl_{c2}'s original vertex order and the vertex $v_{c2,2}$ is used to mark Pl_{c2}'s *location ID*.

We calculate the positions of the vertices $v_{c2,1}$ and $v_{c2,2}$ using the following equations:

$$\begin{cases} \dfrac{v_0^x - v_{c2,1}^x}{v_0^x - v_1^x} = \dfrac{1}{Z+3} \\ \dfrac{v_0^y - v_{c2,1}^y}{v_0^y - v_1^y} = \dfrac{1}{Z+3} \end{cases} \tag{11}$$

and

$$\begin{cases} \dfrac{v_0^x - v_{c2,2}^x}{v_0^x - v_1^x} = \dfrac{m_{p,q}+1}{Z+3} \\ \dfrac{v_0^y - v_{c2,2}^y}{v_0^y - v_1^y} = \dfrac{m_{p,q}+1}{Z+3} \end{cases}. \tag{12}$$

That is, the Euclidean distance between v_0 and v_1 is divided into Z+3 equal intervals, the number of intervals between v_0 and $v_{c2,1}$ is one and the number of intervals between v_0 and $v_{c2,2}$ is equal to $m_{p,q}$ +1.

After inserting the two vertices $v_{c2,1}$ and $v_{c2,2}$ into Pl_{c2}, the marked *complex*2 polyline Pl_{c2}^m has the following characteristics: i) the four vertices it contains are on the same straight line, ii) the number of intervals between the two adjacent vertices at one end of the polyline is one while the number of intervals between the two adjacent vertices at the other end is more than one and iii) the end which the two adjacent vertices containing one interval are at indicates the beginning of the polyline's vertex order.

Because all the four vertices of a marked *complex*2 polyline are used to indicate the polyline's *location ID* and the vertex order, the number of its *mark* vertices is 4.

With these characteristics of a marked *normal* polyline, a marked *complex*1 polyline and a marked *complex*2 polyline, identifying each polyline's original *location ID* and obtaining the original vertex order in the watermark verification stage are possible.

After setting a unique mark on each polyline of G_i, a marked group G_i^m can be obtained.

3.1.3. Watermark Data Generation: To generate the watermark for each marked group, we first divide the vertices into two classes: one is used to generate the watermark (*non-watermark* vertex) and the second one is used to embed the watermark (*watermark* vertex). Let the *watermark* polyline of the marked group G_i^m be the marked polyline Pl^m = $\{v_j | j \in [0, r-1], r \geq \lceil L/c \rceil + 4\}$. Pl^m is also a *normal* polyline, and the first two adjacent vertices and the last two adjacent vertices of it are *mark* vertices. Because the *mark* vertices do not carry watermark bits, the *watermark* vertices should be selected from the rest vertices of Pl^m. Here, we see Pl^m's p ($p = \lceil L/c \rceil$) vertices from v_2 to v_{p+1} as the *watermark* vertices of G_i^m and the rest vertices of G_i^m as the *non-watermark* vertices. Denote the *watermark* vertex set of G_i^m as $V_i^w = \{ v_j(v_j^x, v_j^y) | j \in [0, \lceil L/c \rceil - 1]\}$.

Then, using the *non-watermark* vertices of G_i^m, we generate the watermark H_i by the following function,

$$H_i = grouphash\ (hash\ (I(G_i^m), k, i, M_{in}), L, K), \tag{13}$$

where $I(\cdot)$ is the method for acquiring the RST invariant spatial data of the *non-watermark* vertices and attribution data of all polylines of G_i^m, *hash*$(\cdot)$ represents an existing cryptographic hash method, k is a private key to obtain a random number for the input of *hash*$(\cdot)$, M_{in} denotes the index of the vector map, L is the length of the generated watermark bits, *grouphash*(R, L, K) is a function that returns L bits from a string R in a random fashion under the control of a secret key K, and the bit string $H_i = \{h_j \mid h_j \in \{0,1\}, j \in [0, L-1]\}$ of length L is the generated watermark.

We acquire the RST invariant spatial data of the *non-watermark* vertices for $I(\cdot)$ using the following 4 steps.

*Step*1. Scan the *non-watermark* vertices to get an ordered *non-watermark* vertex list. We traverse the marked polylines of G_i^m from the polyline that has the smallest *location ID* to the polyline that has the biggest *location ID*. Within each polyline, we scan the *non-watermark* vertices from the first vertex to the last vertex. Let N_i^{nw} be the whole number of the *non-watermark* vertices of G_i^m and $V_i^{nw} = \{ v_j(v_j^x, v_j^y) \mid j \in [0, N_i^{nw} - 1]\}$ be the ordered *non-watermark* vertex list.

Step 2. Calculate the center $G_{c,i}$ ($G_{c,i}^x$, $G_{c,i}^y$) of the vertices of V_i^{nw} by

$$G_{c,i} = \frac{1}{N_i^{nw}} \sum_{v_j \in V_i^{nw}} v_j \ . \tag{14}$$

Step 3. Apply the log-polar mapping to the vertices of V_i^{nw} and get a log-radius list $V_i^{nw,r} = \{r_j \mid j \in [0, N_i^{nw} - 1]\}$,

$$r_j = \ln(\sqrt{(v_j^x - G_{c,i}^x)^2 + (v_j^y - G_{c,i}^y)^2}) \ . \tag{15}$$

Step 4. Obtain the RST invariant spatial data $V_i^{nw,s} = \{s_j \mid j \in [0, N_i^{nw} - 2]\}$ of G_i^m by calculating the difference of every two adjacent elements of $V_i^{nw,r}$,

$$s_j = r_j - r_{j+1} \ . \tag{16}$$

The above process of acquiring the RST invariant spatial data for $I(\cdot)$ is shown in Figure 1.

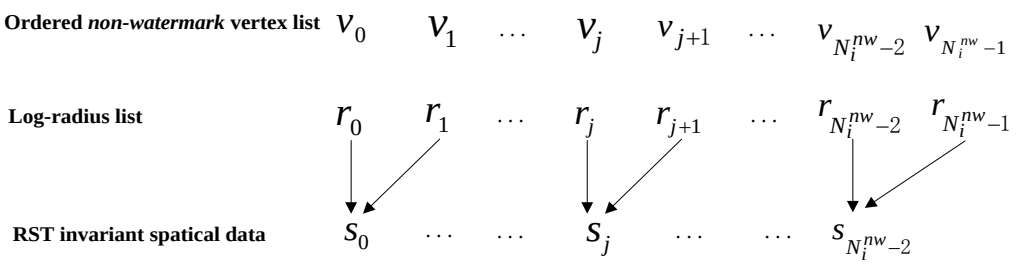

Figure 1. The Process of Acquiring the RST Invariant Spatial Data for I(·)

As the differences of the log-radiuses remain unchanged after RST transformations which will be discussed in Section 4.2, the watermark is invariant to RST. Furthermore, because the feature rearrangement and vertex reversing operations change neither the vertex traversing order nor the data's precision, the watermark is also invariant to these attacks.

3.1.3. Watermark Embedding: Let's assume a watermark $H_i=\{h_j \mid h_j \in \{0,1\}, j \in [0, L-1]\}$ is going to be embedded into the marked group G_i^m, and the *watermark* vertex set of G_i^m which we have obtained in the watermark generation stage is $V_i^w = \{ v_j(v_j^x, v_j^y) \mid j \in [0, \lceil L/c \rceil - 1]$.

We first calculate the watermark sequence $W_i = \{w_i^j \mid j \in [0, \lceil L/c \rceil - 1]\}$ to be embedded using H_i,

$$\begin{cases} w_i^j = h_{(j+1)\times c-1} \times 2^{c-1} + h_{(j+1)\times c-2} \times 2^{c-2} + \cdots + h_{j\times c} \times 2^0, & 0 \le j \le \lceil L/c \rceil - 2 \\ w_i^j = h_{L-1} \times 2^{c-1} + h_{L-2} \times 2^{c-2} + \cdots + h_{L-L\%c} \times 2^{c-L\%c}, & j = \lceil L/c \rceil - 1 \end{cases}. \quad (17)$$

Then, for any element w_i^j of W_i, we embed it into the *watermark* vertex v_j of V_i^w using Eqs. (2)-(4). During embedding, the *reference*2 vertex of G_i^m, the vertex v_j, and the *reference*1 vertex of G_i^m are seen as the *normalization* vertex, the *watermark-embedding* vertex and the *neighboring center*, respectively. The parameter S_w is set as 2^c and the parameter K_w is set using Eq. (8).

After embedding the watermark into each marked group, an embedded vector map M^w is derived.

3.2 Watermark Verification Procedure

The watermark verification process consists of four basic steps: i) Identification of each polyline's *location ID*, ii) Group division, iii) Extraction of group watermarks and iv) Verification of group watermarks.

First of all, we identify the original *location ID* and the vertex order for each polyline. Given a polyline $Pl^w = \{v_i^w(v_i^{w,x}, v_i^{w,y}) \mid i \in [0, r-1], r = 1, 2, 3 \ldots\}$ that contains r vertices, we identify its *location ID* and its vertex order according to four parts, from P1 to P4.

P1 Check if $r \ge 4$. If not, its *location ID* is deemed invalid and it is detected as tampered directly.

P2 Check if $r > 4$. If so, see it as a possible marked *normal* polyline and go through the following 4 steps.

*Step*1. Extract two possible *location ID*s, denoted as $m_{p,q}^1$ and $m_{p,q}^2$, from v_0^w and v_{r-1}^w, respectively. Firstly, by regarding the vertices v_{r-2}^w, v_0^w and v_1^w as the *normalization* vertex, the *watermark-embedding* vertex and the *neighboring center*, respectively, extract $m_{p,q}^1$ from v_0^w using Eq. (2), Eq. (5), Eq.(7) and the inputs (K_w and the total number of polylines Z).

*Step*2. Check if $1 \le m_{p,q}^1 = m_{p,q}^2 - 1 \le Z$ or $1 \le m_{p,q}^2 = m_{p,q}^1 - 1 \le Z$ holds. If not, see its *location ID* as invalid and detect it as tampered directly.

*Step*3. Check if $1 \le m_{p,q}^1 = m_{p,q}^2 - 1 \le Z$ holds. If so, regard $m_{p,q}^1$ as Pl^w's *location ID*, and v_0^w as Pl^w's first vertex.

Step 4. Check if $1 \le m_{p,q}^2 = m_{p,q}^1 - 1 \le Z$ holds. If so, regard $m_{p,q}^2$ as Pl^w's *location ID*, and v_{r-1}^w as Pl^w's first vertex.

P3 Check if the 4 vertices Pl^w ($r = 4$) contains are on the same straight line. If so, see it as a possible marked *complex*2 polyline, and go through the following 4 steps.

Step 1. Divide the Euclidean distance between v_0^w and v_3^w into Z+3 equal intervals. Denote the number of intervals between v_0^w and v_1^w as N_f, and the number of intervals between v_2^w and v_3^w as N_e.
Step 2. Check if N_f=1 and N_e > 1 or N_e =1 and N_f > 1 hold. If not, see its *location ID* as invalid and detect it as tampered directly.
Step 3. Check if N_f=1 and N_e > 1 hold. If so, regard v_0^w as the first vertex of Pl^w and calculate its *location ID m* by the following:

$$\begin{cases} \dfrac{v_0^{w,x}-v_2^{w,x}}{v_0^{w,x}-v_3^{w,x}}=\dfrac{m+1}{Z+3} \\ or \\ \dfrac{v_0^{w,y}-v_2^{w,y}}{v_0^{w,y}-v_3^{w,y}}=\dfrac{m+1}{Z+3} \end{cases}. \quad (18)$$

Step 4. Check if N_e =1 and N_f > 1 hold. If so, regard v_3^w as the first vertex of Pl^w and calculate Pl^w's *location ID m* with the following equation,

$$\begin{cases} \dfrac{v_3^{w,x}-v_1^{w,x}}{v_3^{w,x}-v_0^{w,x}}=\dfrac{m+1}{Z+3} \\ or \\ \dfrac{v_3^{w,y}-v_1^{w,y}}{v_3^{w,y}-v_0^{w,y}}=\dfrac{m+1}{Z+3} \end{cases}. \quad (19)$$

P4 Check if the first 3 adjacent vertices (v_0^w, v_1^w and v_2^w) or the last 3 adjacent vertices (v_1^w, v_2^w and v_3^w) of Pl^w (r = 4) are on the same straight line. If not, see the polyline as a possible marked *normal* polyline, and obtain its original *location ID* and vertex order by going through the 4 steps of P2; otherwise, see it as a possible marked *complex*1 polyline, and go through the following 2 steps.
Step 1. Check if v_0^w, v_1^w and v_2^w are in the same line. If so, regard v_0^w as the first vertex of Pl^w and calculate Pl^w's *location ID m* by

$$\begin{cases} \dfrac{v_0^{w,x}-v_1^{w,x}}{v_0^{w,x}-v_2^{w,x}}=\dfrac{m}{Z+1} \\ or \\ \dfrac{v_0^{w,y}-v_1^{w,y}}{v_0^{w,y}-v_2^{w,y}}=\dfrac{m}{Z+1} \end{cases}. \quad (20)$$

Step 2. Check if v_1^w, v_2^w and v_3^w are in the same line. If so, regard v_3^w as the first vertex of and calculate Pl^w's *location ID m* with the following equation,

$$\begin{cases} \dfrac{v_3^{w,x}-v_2^{w,x}}{v_3^{w,x}-v_1^{w,x}}=\dfrac{m}{Z+1} \\ or \\ \dfrac{v_3^{w,y}-v_2^{w,y}}{v_3^{w,y}-v_1^{w,y}}=\dfrac{m}{Z+1} \end{cases}. \quad (21)$$

Later, the original polylines of each group can be obtained according to their *location ID*s. Assuming a marked polyline's *location ID* is *m*, we can get its group index *i* ($0 \le i \le N_g - 1$) and its location *j* ($1 \le j \le n$) in the corresponding watermarked group G_i^w by

$$\begin{cases} i = (m-1)/n \\ j = m - i \times n \end{cases}. \quad (22)$$

Then, using Eq. (2) and Eq. (5), we extract the embedded watermark from the *watermark* vertices for each group with the input K_w and the parameter $S_w = 2^c$. Assume the *watermark* vertex set of G_i^w is $V_i^{w\prime} = \{ v_j' \mid j \in [0, \lceil L/c \rceil - 1]$, the extracted watermark sequence as $W_i' = \{w_i^{j\prime} \mid j \in [0, \lceil L/c \rceil - 1]\}$, and the *reference*1 vertex and the *reference*1 vertex of G_i^w are v_{r1}' and v_{r2}', respectively. For any *watermark* vertex v_j' of $V_i^{w\prime}$, we extract an element of W_i' ($w_i^{j\prime}$) from it by regarding v_{r2}', v_j', and v_{r1}' as the *normalization* vertex, the *watermark-embedding* vertex and the *neighboring center*, respectively. By sequentially concatenating the elements of W_i', we can get the extracted watermark H_i^1 of G_i^w.

After that, we generate each group's watermark using the watermark generation method. The polyline order and the vertex order we used here are obtained in the step of identifying each polyline's original *location ID*.

At the last step, whether the group has been tampered can be judged by the mismatch between the extracted watermark and the watermark derived from the received content. A group is deemed authentic if the two watermarks are equal; otherwise it is seen as tampered. The inserted vertices used to indicate the *location ID* and the vertex order for *complex*1 polylines and *complex*2 polylines can be deleted for saving storage space.

4. Results and Analysis

4.1 Experimental Results

We ran experiments on a PC with CPU 4400+ 2.31GHz, RAM 3G, WinXP Professional, ArcMap Version9.2, Map Objects 2.4 and Visual C++6.0. In the experiment, 50 different 2D vector maps were used as the covers. As shown in Fig. 2, six of them are a coastline map of Taylor Rookery [22], a river map [23], a coastline map of Windmill islands [24], a land map [23], a Roraima State map [23] and a lake map of SR41-42 Northern Prince Charles Mountains [25]. Table 1 lists some basic properties of the six vector maps, including the feature type, the number of vertices/features, the scale and the precision tolerance τ. During watermark embedding, the MD5 algorithm was employed as the hash function and the parameters were chosen as follows: watermark length $L = 128$, group size $n = 3$, and the number of watermark bits a vertex carries $c = 8$.

In the first test case, we demonstrated the invisibility of our scheme. The average embedding distortion *d* and the maximum distortion [18] *Maxd* were exploited to measure the embedded vector map objective quality,

$$\begin{aligned} d(M, M^w) &= \frac{1}{V^M} \sum_{i=1}^{V^M} \left\| v_i - v_i^w \right\| \\ Maxd(M, M^w) &= \max\left(\left\| v_i - v_i^w \right\| \right), \ (i = 1,2,\ldots,V^M) \end{aligned}. \quad (23)$$

where v_i the and v_i^w are the corresponding vertices in the original vector map (M) and the watermarked vector map (M^w) and V^M denotes the total number of vertices in the vector map M. The vector maps in Figure 2 were watermarked by the proposed technique

yielding the watermarked versions seen in Figure 3. The experimental results are listed in Table 2. Due to the configuration of the embedding parameter K_w (Eq. (8)), the introduced distortions do not exceed the precision τ, and hence the validity of the map data is guaranteed.

The robustness against rotation, uniform scaling and translation was illustrated in the second test case. We rotated the watermarked river map shown in Figure 3(b) by different angles, scaled it with different factors and translated it with different $\triangle x$ and $\triangle y$ in the x and y axes, respectively. Since the authentication watermark is invariant to rotation, uniform scaling and translation operations, the manipulated vector map has passed the authentication in each experiment, as shown in Table 3, Table 4 and Table 5.

Figure 2. Test 2D Vector Maps: (a) Coastline Map of Taylor Rookery, (b) river map, (c) Coastline Map of Windmill Islands, (d) Land Map, (e) Roraima State Map and (f) Lake Map of SR41-42 Northern Prince Charles Mountains

Table 1. Properties of Original Vector Maps

2D vector map	Feature type	Vertices/features	Scale	τ (m)
Coastline map of Taylor Rookery	polyline	4279/18	1:5000	0.5
River map	polyline	23854/1084	1:25000	2.5
Coastline map of Windmill islands	polyline	38082/496	1:50000	5
Land map	polygon	47170/35	1:100000	10
Roraima State map	polygon	574679/202	1:250000	25
Lake map of SR41-42 Northern Prince Charles Mountains	polygon	3138/55	1:1000000	100

In the third test case, we demonstrated the tamper detection and localization ability of the proposed algorithm. The 2D vector map we used was the river map as shown in Figure 2(b). Figure 4 shows the original map at different stages of watermarking.

The original river map in Figure 4(a) was watermarked by the proposed algorithm yielding the watermarked map seen in Figure 4(b). Then, the watermarked map was manipulated using ArcMap to yield the map in Figure 4(c). In particular, 2 vertices were added to region 'A', 2 vertices in region 'B' were modified and 6 vertices were deleted from region 'C'. In the third step, integrity and authenticity of the manipulated vector map were tested using the watermark verification algorithm. Output of the watermark verification step is seen in Figure 4(d). Red features indicate the located suspicious polyline groups.

From Figure 4(d), we can find that the red features are exactly where the tampering operations happen.

Table 2. The Maximum Distortion *Maxd* and the Average Distortion *d* Introduced by Watermark Embedding

2D vector map	*Maxd* (m)	*d* (m)
Coastline map of Taylor Rookery	0.25483	0.00056
River map	1.96858	0.05261
Coastline map of Windmill islands	1.67688	0.00203
Land map	0.27600	0.00007
Roraima State map	0.72412	0.00006
Lake map of SR41-42 Northern Prince Charles Mountains	3.77436	0.05805

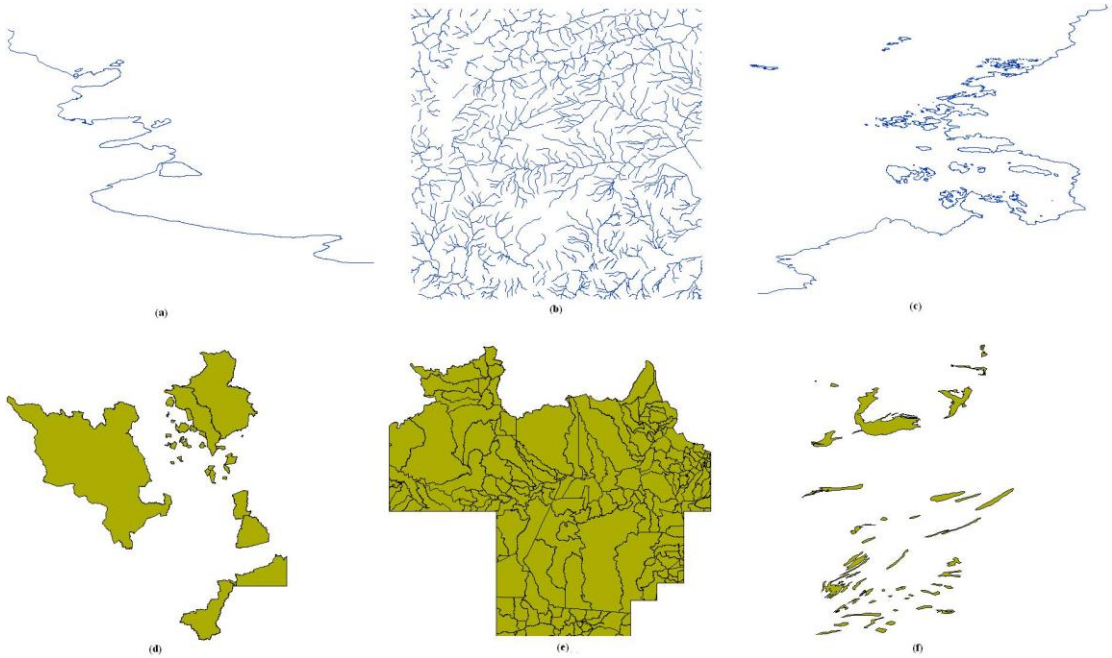

Figure 3. The Watermarked 2D Vector Maps of Figure 2

Table 3. Experiment Results of Rotation

Rotation angle (degree)	30	60	90	120	150	180	240	300
Authentication results	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass

Table 4. Experiment Results of Uniform Scaling

Scale factor	0.25	0.5	2.5	4.0	5.5	7.5	9.5
Authentication results	Pass	Pass	Pass	Pass	Pass	Pass	Pass

Table 5. Experiment Results of Translation

Distance (△x m, △y m)	(– 1.2, 2.3)	(4.2, 5.6)	(2.6, – 7.9)	(– 6.5, – 4.8)
Authentication results	Pass	Pass	Pass	Pass

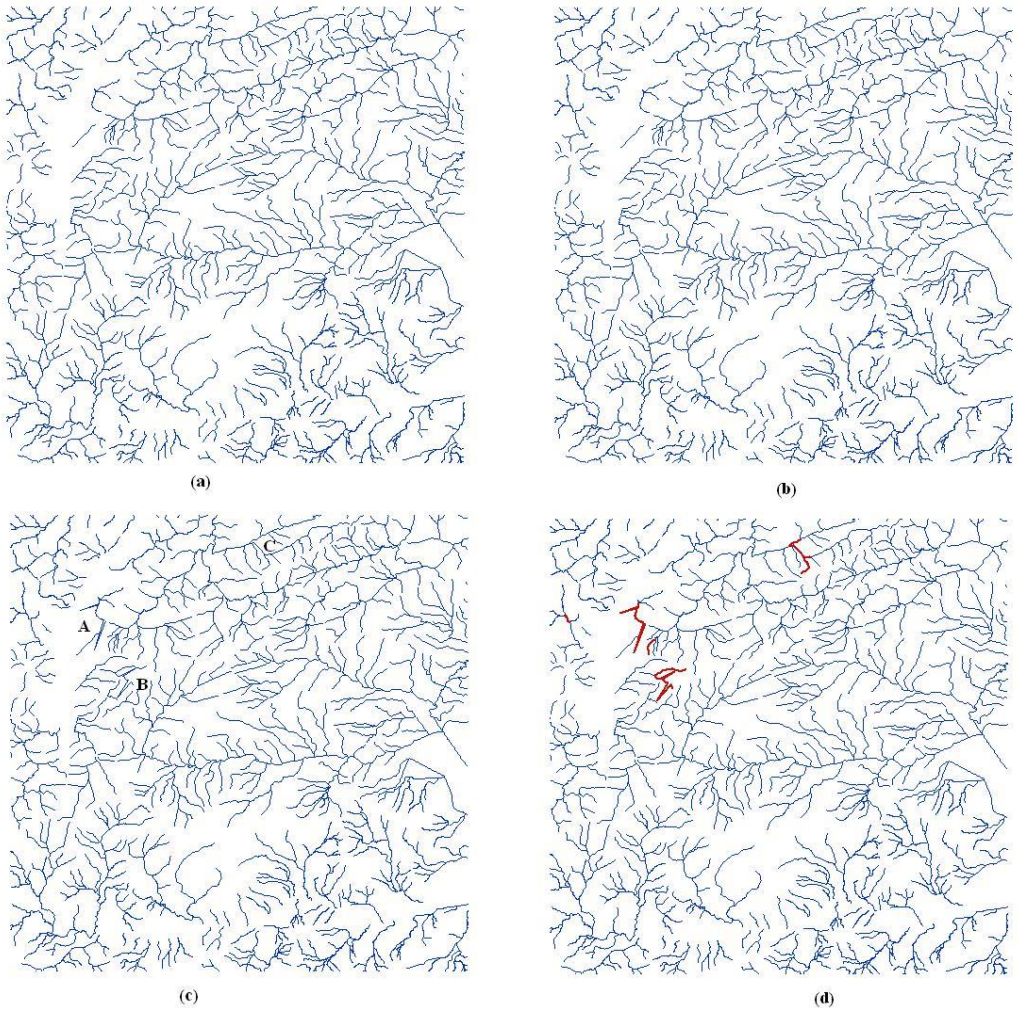

Figure 4. (a) The Original River Map, (b) the Watermarked Map, (c) the Same View after the Watermarked Map has been Modified and (d) the Tamper Localization Results: the Red Features Represent the Suspicious Feature Groups

4.2 Discussion of RST Invariance Property

In this subsection, we discuss the RST invariance property of the proposed algorithm. In the following, we assume that group G^w is the watermarked version of group G, $G^{w'}$ is a rotated/scaled/translated version of G^w, H is the watermark that G^w carries, H^w is the watermark extracted from $G^{w'}$, $H^{w'}$ is the watermark derived from $G^{w'}$, $V^{nw} = \{ v_j(v_j^x, v_j^y) | j \in [0, N^{nw} - 1]\}$ with N^{nw} vertices is the ordered *non-watermark* vertex list of G, $G_c(G_c^x,$

G_c^y)is the center of the vertices of V^{nw} and $V^{nw,s} = \{s_j | j \in [0, N^{nw} - 2]\}$ is the RST invariant spatial data of G. As *non-watermark* vertices do not carry watermark bits, V^{nw} is also the ordered *non-watermark* vertex list of G^w.

Since the watermark and each polyline's *location ID* are embedded using Chou and Tseng's method [21], the embedded watermark H remains unchanged after a rotated/scaled/translated operation, *i.e.*, $H = H^w$.

4.2.1 Rotation Invariance: Given a *non-watermark* vertex $v_j(v_j^x, v_j^y)$ $(0 \le j \le N^{nw} - 1)$ of V^{nw}, a rotation by an angle ρ leads to a new *non-watermark* vertex $v_j'(v_j^{x'}, v_j^{y'})$ and a new center $G_c'(G_c^{x'}, G_c^{y'})$ of the ordered *non-watermark* vertex list V^{nw}

,

$$\begin{cases} (v_j^x', v_j^y') = (v_j^x \times \cos\rho + v_j^y \times \sin\rho,\ -v_j^x \times \sin\rho + v_j^y \times \cos\rho) \\ (G_c^x', G_c^y') = (G_c^x \times \cos\rho + G_c^y \times \sin\rho,\ -G_c^x \times \sin\rho + G_c^y \times \cos\rho) \end{cases} \quad (24)$$

After applying the log-polar mapping to $v_j'(v_j^{x'}, v_j^{y'})$, the new log-radius r_j' becomes

$$\begin{aligned} r_j' &= \ln\left(\sqrt{(v_j^x' - G_c^x')^2 + (v_j^y' - G_c^y')^2}\right) \\ &= \ln\left(\sqrt{(v_j^x - G_c^x)^2 + (v_j^y - G_c^y)^2}\right) = r_j \end{aligned}, \quad (25)$$

which means the watermark derived from the rotated version is the same as the embedded watermark, *i.e.*, $H = H^{w'}$. Thus, $H^{w'} = H^w$, and the algorithm is invariant to rotation.

4.2.2 Scaling Invariance: Given two adjacent *non-watermark* vertices $v_j(v_j^x, v_j^y)$ and $v_{j+1}(v_{j+1}^x, v_{j+1}^y)$ $(0 \le j \le N^{nw} - 2)$ of V^{nw}, after a scaling operation by a factor s $(s > 0)$, the new *non-watermark* vertices $v_j'(v_j^{x'}, v_j^{y'})$ and $v_{j+1}'(v_{j+1}^{x'}, v_{j+1}^{y'})$, and the new center $G_c'(G_c^{x'}, G_c^{y'})$ of the ordered *non-watermark* vertex list V^{nw} become

$$\begin{cases} (v_j^x', v_j^y') = (v_j^x \times s, v_j^y \times s) \\ (v_{j+1}^x', v_{j+1}^y') = (v_{j+1}^x \times s, v_{j+1}^y \times s) \\ (G_c^x', G_c^y') = (G_c^x \times s, G_c^y \times s) \end{cases}. \quad (26)$$

After applying the log-polar mapping to $v_j'(v_j^{x'}, v_j^{y'})$ and $v_{j+1}'(v_{j+1}^{x'}, v_{j+1}^{y'})$, the new log-radiuses r_j' and r_{j+1}' become

$$\begin{cases} r_j' = \ln\left(\sqrt{(v_j^x' - G_c^x')^2 + (v_j^y' - G_c^y')^2}\right) \\ \quad = \ln s + \ln\left(\sqrt{(v_j^x - G_c^x)^2 + (v_j^y - G_c^y)^2}\right) \\ r_{j+1}' = \ln\left(\sqrt{(v_{j+1}^x' - G_c^x')^2 + (v_{j+1}^y' - G_c^y')^2}\right) \\ \quad = \ln s + \ln\left(\sqrt{(v_{j+1}^x - G_c^x)^2 + (v_{j+1}^y - G_c^y)^2}\right) \end{cases}. \quad (27)$$

Then, the new j-th element s_j' of $V^{nw,s}$ becomes

$$s_j' = r_j' - r_{j+1}' = r_j - r_{j+1} = s_j. \quad (28)$$

Thus, $H = H^{w'} = H^w$, and the algorithm is invariant to scaling.

4.2.3 Translation Invariance: Given a *non-watermark* vertex $v_j(v_j^x, v_j^y)$ $(0 \le j \le N^{nw} - 1)$ of V^{nw}, a translation by $\triangle x$ and $\triangle y$ in the x and y axes, respectively, leads to a new *non-watermark* vertex $v_j'(v_j^{x'}, v_j^{y'})$ and a new center $G_c'(G_c^{x'}, G_c^{y'})$ of the ordered *non-watermark* vertex list V^{nw},

$$\begin{cases}(v_j^{x}', v_j^{y}') = (v_j^x + \Delta x, v_j^y + \Delta y) \\ (G_c^{x}', G_c^{y}') = (G_c^x + \Delta x, G_c^y + \Delta y)\end{cases}. \tag{29}$$

After applying the log-polar mapping to $v_j'(v_j^{x'}, v_j^{y'})$, the new log-radius r_j' becomes

$$\begin{aligned} r_j' &= \ln\left(\sqrt{(v_j^{x}' - G_c^{x}')^2 + (v_j^{y}' - G_c^{y}')^2}\right) \\ &= \ln\left(\sqrt{(v_j^x - G_c^x)^2 + (v_j^y - G_c^y)^2}\right) = r_j \end{aligned}. \tag{30}$$

We also have $H = H^{w'} = H^w$. Thus, the algorithm is invariant to translation.

4.3 Discussion of Localization Accuracy

To measure the tamper localization ability of the algorithm in features, the metrics β [18] which represents the number of polylines detected as tampered after malicious attacks is used. The expectation of β, denoted as $Ex(\beta)$, was calculated to compare the localization accuracy of the proposed method with the ones reported in [17-18].

In [17], Zheng et al. divide the features into different groups according to the number of vertices a feature contains. However, the number of vertices within each feature is difficult to evaluate, which make the calculation of $Ex(\beta)$ very hard. For simplicity, we assume that the vector map, which consists of Z polylines, is divided into N_g groups, each of which contains n features ($Z = n \times N_g$), the probability that the added/deleted/modified feature belongs to the i-th ($0 \le i \le N_g - 1$) group is $1/N_g$, and adding/deleting/modifying a vertex or modifying a feature does not change the grouping. Then, we obtain the lower limit of $Ex(\beta)$ after a vertex addition/deletion/modification attack and a feature modification attack. Since after a feature rearrangement attack, the probability that D ($1 \le D \le N_g$) groups are tampered is greater than the probability that D –1 groups are tampered, we assume the probability that D groups are tampered is $1/ N_g$. Thus, the lower limit of $Ex(\beta)$ after a feature rearrange attack is obtained.

When calculating $Ex(\beta)$ for the proposed method in [18], we assume the situation that the modified feature is divided into another group occurs with a probability close to zero, and the probability that the added feature is regarded as a valid feature is 1/2.

Table 6. Tamper Localization Accuracy of Different Methods

Attacks	$Ex(\beta)$ of the method in [17]	$Ex(\beta)$ of the method in [18]	$Ex(\beta)$ of the proposed method
Vertex addition/deletion/modification	n	$\approx n$	$\approx n$
Feature addition	$(n + Z)/2+1$	$1+n/2$	$1+n/2$
Feature deletion	$(n + Z)/2-1$	$n - 1$	$n - 1$
Feature modification	n	$\approx n$	$\approx n$
Feature rearrangement	$(n + Z)/2$	0	0
Vertex reversing	n	0	0
Rotation/scaling/translation	Z	Z	0

From the results in Table 6, we can find that the proposed method has better localization capability than Zheng *et al.'s* method [17]. Besides, the proposed algorithm performs better in terms of RST invariance than the methods in [17-18].

5. Conclusions

In this paper, we describe a RST invariant fragile watermarking scheme for 2D vector map authentication based on Chou and Tseng's watermark embedding method [21]. By folding the hash results of the differences of the log-radiuses to generate the authentication watermark, the proposed method is tolerant to RST transformations, but sensitive to malicious attacks. As a unique mark indicating the original location of each feature is embedded into the feature itself, the algorithm can precisely locate tampered groups after malicious attacks. Meanwhile, with the vertex scanning order indicated by each feature's mark, this scheme is immune to feature rearrangement and vertex reversing operations.

Our RST invariant watermark generation method can be integrated with other RST invariant watermarking schemes in order to obtain both the sensitivity to malicious attacks and the RST invariance property. Besides, the idea of setting a unique mark on each feature can also be integrated with other RST invariant watermarking schemes for good tamper localization capability.

References

[1] C. López, "Watermarking of digital geospatial datasets: A review of technical, legal and copyright issues", International Journal of Geographical Information Science, vol. 16, no. 6, (**2002**).
[2] X. Niu, C. Shao and X. Wang, "A survey of digital vector map watermarking", International Journal of Innovative Computing, Information and Control, vol. 2, no. 6, (**2006**).
[3] J. Lafaye, J. Béguec, D. Gross-Amblard and A. Ruas, "Blind and squaring-resistant watermarking of vectorial building layers", Geoinformatica, vol. 16, no. 2, (**2012**).
[4] S.-H. Lee and K.-R. Kwon, "Vector watermarking scheme for GIS vector map management", Multimedia Tools and Applications, vol. 63, no. 3, (**2013**).
[5] E. Horness, N. Nikolaidis and I. Pitas, "Blind city maps watermarking utilizing road width information", 15th European Signal Processing Conference, (**2007**) September 3-7; Poznan, Poland.
[6] V. R. Doncel, N. Nikolaidis and I. Pitas, "An optimal detector structure for the Fourier descriptors domain watermarking of 2D vector graphics", IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 5, (**2007**).
[7] K. Jiang, K. Q. Zhu, Y. Huang and X. Ma, "Watermarking road maps against crop and merge attacks", Proceedings of the first ACM workshop on Information hiding and multimedia security, (**2013**).
[8] R. Ohbuchi, H. Ueda and S. Endoh, "Robust watermarking of vector digital maps", Proceedings of the IEEE International Conference on Multimedia and Expo, (**2002**).
[9] H. Gou and M. Wu, "Data hiding in curves with application to fingerprinting maps", IEEE Transactions on Signal Processing, vol. 53, no. 10, (**2005**).
[10] X. Wang, C. Shao, X. Xu and X. Niu, "Reversible data-hiding scheme for 2-D vector maps based on difference expansion", IEEE Transactions on Information Forensics and Security, vol. 2, no. 3, (**2007**).
[11] M. Voigt, B. Yang and C. Busch, "Reversible watermarking of 2D-vector data", Proceedings of the Multimedia and Security Workshop, (**2004**).
[12] L. Cao, C. Men and R. Ji, "High-capacity reversible watermarking scheme of 2D-vector data", Signal, Image and Video Processing, (**2014**).
[13] N. Wang, H. Zhang and C. Men, "A high capacity reversible data hiding method for 2D vector maps based on virtual coordinates", Computer-Aided Design, vol. 47, (**2014**).
[14] J. Fridrich, M. Goljan and R. Du, "Invertible authentication", Proceedings of SPIE, (**2001**) August 1.
[15] C. Shao, X. Wang and X. Xu, "Security issues of vector maps and a reversible authentication scheme", Doctoral Forum of China, (**2005**) July; Beiging, China.
[16] L. Zheng and F. You, "A fragile digital watermark used to verify the integrity of vector map", International Conference on E-Business and Information System Security (EBISS), (**2009**) May 23-24; Wuhan China.
[17] L. Zheng, Y. Li, L. Feng and H. Liu, "Research and implementation of fragile watermark for vector graphics", International Conference on Computer Engineering and Technology, (2010) April 16-18; Chengdu China.
[18] N. Wang and C. Men, "Reversible fragile watermarking for 2-D vector map authentication with localization", Computer-Aided Design, vol. 44, no. 4, (**2012**).

[19] N. Wang and C. Men, "Reversible fragile watermarking for locating tampered blocks in 2D vector maps", Multimedia Tools and Applications, vol. 67, no. 3, (**2013**).
[20] ESRI shapefile technical description, http://www.esri.com/library/whitepapers, 1998, Accessed (**2014**) April 9.
[21] C.-M. Chou and D.-C.Tseng, "Affine-transformation-invariant public fragile watermarking for 3D model authentication", IEEE Computer Graphics and Application, vol. 29, no. 2, (**2009**).
[22] U. Harris and T. Rookery, "1:5000 Topographic GIS Dataset. Australian Antarctic Data Centre – CAASM Metadata.
[23] http://gcmd.nasa.gov/KeywordSearch/Metadata.do?Portal=amd_au&MetadataView=Full&MetadataType=0&KeywordPath=&OrigMetadataNode=AADC&EntryId=Tayl5k, (1999, updated 2008)
[24] http://www.ibge.gov.br/english/geociencias/default_prod.shtm, (**2012**).
[25] U. Harris, "Windmill Islands 1:50000 Topographic GIS Dataset", Australian Antarctic Data Centre - CAASM Metadata.
[26] http://gcmd.nasa.gov/KeywordSearch/Metadata.do?Portal=amd_au&MetadataView=Full&MetadataType=0&KeywordPath=&OrigMetadataNode=AADC&EntryId=Wind50k. (1999, updated)
[27] U. Harris. SR41-42 Northern Prince Charles Mountains - 1:1 Million Topographic GIS Dataset," Australian Antarctic Data Centre - CAASM Metadata. http://gcmd.nasa.gov/KeywordSearch/Metadata.do?Portal=amd_au&MetadataView=Full&MetadataType=0&KeywordPath=&OrigMetadataNode=AADC&EntryId=SR41-42, (1999, updated 2008)

