PAPER

Robust Hashing of Vector Data Using Generalized Curvatures of Polyline*

Suk-Hwan LEE[†a)], ***Member***, **Seong-Geun KWON**[††b)], ***and*** **Ki-Ryong KWON**[†††c)], ***Nonmembers***

SUMMARY With the rapid expansion of vector data model application to digital content such as drawings and digital maps, the security and retrieval for vector data models have become an issue. In this paper, we present a vector data-hashing algorithm for the authentication, copy protection, and indexing of vector data models that are composed of a number of layers in CAD family formats. The proposed hashing algorithm groups polylines in a vector data model and generates group coefficients by the curvatures of the first and second type of polylines. Subsequently, we calculate the feature coefficients by projecting the group coefficients onto a random pattern, and finally generate the binary hash from binarization of the feature coefficients. Based on experimental results using a number of drawings and digital maps, we verified the robustness of the proposed hashing algorithm against various attacks and the uniqueness and security of the random key.

key words: *vector data model, content hashing, design drawing, digital map, curvature*

1. Introduction

Vector data models have been widely applied to various types of content, including digital maps for GIS (global information systems) [1] and CAD (computer-aided design) [2] drawings for architectures, cars, shipbuilding, and IT (information technology) hardware designs. In response to the demand for greater security with the development of such content based on vector data models, many watermarking schemes have been presented for the copyright protection of CAD drawing designs [3], [4], GIS digital maps [5]–[13], vector graphics [14], and 3D model [15]–[18]. On the other hand, E.J. Delp [19] suggested the importance of content authentication, rather than copyright protection, along with changes in the business model, as an approach toward multimedia security in the future. Content-based hashing is a typical technique for content authentication. The cryptographic hash functions cannot be used on digital multimedia because they are very sensitive to every bit of digital data. Many researchers have presented robust and secure hash functions for images [20], [21], videos [22], [23], and 3D models [24]–[26]. However, these functions cannot be used for vector data models because the data structure for the latter is different. Furthermore, there has been little interest on hash functions for vector data models.

Like watermarking, content-based hashing must be designed according to the data structure of the content. Images are represented by pixel arrays of fixed position and resolution, and videos are represented by a number of image frames. Many image and video hashing schemes have been proposed in the frequency domains [20]–[23]. 3D polygonal models are represented by a number of vertices and connectivity that can be decreased by simplification or increased by subdivision. 3D model hashing has been proposed by using the shape features [24]–[26]. Vector data models consist of primitive entities, such as points, polylines, polygons, circles, and text, to construct more complex objects. Especially, vector data models by AutoCAD DXF/DWG family formats are designed by layers that have a number of primitive entities. Therefore, the hash function must be suitable to these vector data models.

In this paper, we present a robust and secure hashing algorithm for vector data models of design drawings and digital maps in the AutoCAD DXF/DWG family of formats. The main features of our algorithm are as follows. First, we use the generalized curvature of a polyline as the base value of our algorithm. The generalized curvature is invariant to re-parameterization and Euclidean transformation and also to geometric transformation with shape preservation. Many kinds of attacks on vector data models are based on re-parameterization, similarity transformation, and geometric transformation. Therefore, this is very effective in terms of robustness. Second, we select primary layers with high density and clustered polylines using curve energies. A few layers in many models have most of the primitives and they are considered as primary layers. Our algorithm extracts feature values from them for hashing. The objective of the clustering based on bending energy is to provide robustness to data rearrangement and geometric transformation, and also to improve the security, since the clustering depends on the initial parameters. Thirdly, we generate the binary hash by thresholding feature values of matrix form. The feature values are obtained by the combination of the random values and the group values of the first and second

Manuscript received June 30, 2011.
Manuscript revised November 13, 2012.

[†]The author is with the Department of Information Security, Tongmyong University, Korea.
[††]The author is with the Department of Electronics Engineering, KyungIl University, Korea.
[†††]The author is with the Department of IT Convergence and Application Engineering, Pukyong National University, Korea.

*This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MEST) (KRF-2009-0071269) and the framework of international cooperation program managed by National Research Foundation of Korea (2012K2A1A2032979).

a) E-mail: skylee@tu.ac.kr
b) E-mail: sgkwon@kiu.ac.kr
c) E-mail: krkwon@pknu.ac.kr (Corresponding author)

DOI: 10.1587/transinf.E96.D.1105

curvatures. Therefore, they have security as well as the robustness to attacks.

Based on experimental results, we verified that the proposed hash function shows robustness against various geometric attacks available to vector data, uniqueness in terms of the key and model, and high security based on differential entropy.

The rest of paper is organized as follows. We introduce the vector data-hashing algorithm and the theory of generalized curvature in Sect. 2. We then explain the proposed hashing scheme in Sect. 3. In Sect. 4, we evaluate the robustness, uniqueness, and security of the proposed scheme. Finally, we conclude our paper and discuss our future work in Sect. 5.

2. Related Works

2.1 Vector Data Hashing

Content-based hashing algorithms [20]–[26] that generate a binary hash by combining the feature values of content to the random values have been presented for the robustness, security, and uniqueness, which are the fundamental requirements of hashing. Given that feature extraction is the main process for robustness, it must be designed by considering the data structure of the content. Robustness guarantees that the hash is kept to the attack that preserves the shape quality. The kinds of attacks are different for images, videos, 3D models, and vector data models. The main difference between vector data hashing and image/3D model hashing is how the feature vector can be extracted to be robust. The layers and objects in vector data models can be easily modified through CAD/GIS editing tools. Therefore, vector data hashing must be robust against layer and object modifications. The security technique aims to guarantee that the hash cannot be estimated without knowledge of a key. Security can be improved using a random or permutation key in the process of feature extraction and binarization or using random quantization. Uniqueness aims to guarantee that the hash generated from any model or key is unique. Thus, hashes generated from the same model with different keys are unrelated. Similarly, hashes generated from different models with the same key are unrelated. Uniqueness is affected by feature extraction and the combination of feature and random vectors.

Vector data models in AutoCAD DXF/DWG family formats, which are very popular formats, are designed by a number of layers that consist of primitive entities such as points, lines, faces, circles, and text. The layers among those with polylines or single lines are used as the target layers for hash extraction. The feature vector for a vector data model can be extracted by the geometric feature in each layer. Layers and primitives are arrayed on their indices that can be easily rearranged. Thus, the array order or index of a layer or primitive is variable, unlike the pixel array of an image. Therefore, the vector data-hashing algorithm must be able to extract a feature vector that is robust to the rearrangement of layers or primitives and other geometric modifications. The proposed hashing scheme extracts the hash feature vector using the curvatures of the polylines, which are the main primitive entity in design drawings and digital maps.

2.2 Generalized Curvature of Line Curve

We use the first and second curvatures, χ_1, χ_2, of polyline for the hash feature vector because it is invariant to line reparameterization and Euclidean transformation. This property is very important for the robustness of hash.

The following is a brief review of the theory of generalized curvature [27], [28]. Let n and r be a natural number, $\mathbf{I}$ be a non-empty interval of real numbers, and t be any real number in $\mathbf{I}$. Then, a parametric curve Υ is defined as a vector-valued function, $\Upsilon : \mathbf{I} \rightarrow \mathbf{R}^n$, that is r times continuously differentiable in $\mathbf{R}^n$, which is regular of order m; $\{\Upsilon'(t), \Upsilon''(t), \cdots, \Upsilon^{(m)}(t)\}, m \leq r, t \in \mathbf{I}$. The Frenet frame of a parametric curve Υ is the set of m orthonormal vectors, $\{\mathbf{e}_1(t), \cdots, \mathbf{e}_m(t)\}$, which are called Frenet vectors. The Frenet frame is a moving reference frame of m orthonormal vectors, such as the curvature or torsion at each point of $\Upsilon(t)$, that are used for describing a curve locally. Each orthonormal vector $\mathbf{e}_j(t)$ can be reconstructed from derivatives of $\Upsilon(t)$ using the Gram-Schmidt orthogonalization algorithm; $\mathbf{e}_j(t) = \bar{\mathbf{e}}_j(t)/\|\bar{\mathbf{e}}_j(t)\|$ where $\bar{\mathbf{e}}_j(t) = \Upsilon^{(j)}(t) - \sum_{i=1}^{j-1} < \Upsilon^{(j)}(t), \mathbf{e}_i(t) > \mathbf{e}_i(t)$ and $\mathbf{e}_1(t) = \Upsilon'(t)/\|\Upsilon'(t)\|$. The generalized curvature is defined as the real function, $\chi_i(t) =< \mathbf{e}_i'(t), \mathbf{e}_{i+1}(t) > /\|\Upsilon'(t)\|$. The Frenet frame and the generalized curvature have differential geometric properties of the curve, which are invariant after reparameterization.

In this paper, we obtain the first and second curvatures of a discrete polyline using the Frenet frames. Given a polyline $\mathbf{p}$ with $n+1$ vertices, $\{\mathbf{v}_0, \cdots, \mathbf{v}_n\}$, which is piecewise continuous, the Frenet vectors $\mathbf{e}_j[k]$ for vertices $\mathbf{v}_k$ are

$$\mathbf{e}_j[k] = \frac{\bar{\mathbf{e}}_j[k]}{\|\bar{\mathbf{e}}_j[k]\|}, k \in [0, n] \tag{1}$$

where $\mathbf{e}_1[k] = \Upsilon'[k]/\|\Upsilon'[k]\|$ and $\bar{\mathbf{e}}_j[k] = \Upsilon^{(j)}[k] - \sum_{i=1}^{j-1} < \Upsilon^{(j)}[k], \mathbf{e}_i[k] > \mathbf{e}_i[k]$. The first Frenet vector $\mathbf{e}_1[k]$ is the unit tangent vector that is defined at each point $\mathbf{v}_k$ of a parametric curve Υ; the second Frenet vector $\mathbf{e}_2[k]$ is the unit curvature vector or the unit normal vector that describes the deviance of the curve from a straight line. The two vectors $\mathbf{e}_1[k]$ and $\mathbf{e}_2[k]$ at a point $\mathbf{v}_k$ define the osculating plane at this point. The third Frenet vector $\mathbf{e}_3[k]$ is the binormal vector, which is always orthogonal to $\mathbf{e}_1[k]$ and $\mathbf{e}_2[k]$. We obtain the first curvature $\chi_1[k]$ using $\mathbf{e}_1[k]$ and $\mathbf{e}_2[k]$, and the second curvature $\chi_2[k]$ using $\mathbf{e}_2[k]$ and $\mathbf{e}_3[k]$.

$$\chi_1[k] = \frac{< \mathbf{e}_1'[k], \mathbf{e}_2[k] >}{\|\Upsilon'[k]\|}, \chi_2[k] = \frac{< \mathbf{e}_2'[k], \mathbf{e}_3[k] >}{\|\Upsilon'[k]\|} \tag{2}$$

We then extract the feature value of a polyline by the two curvatures, $\chi_1[k]$ and $\chi_2[k]$.

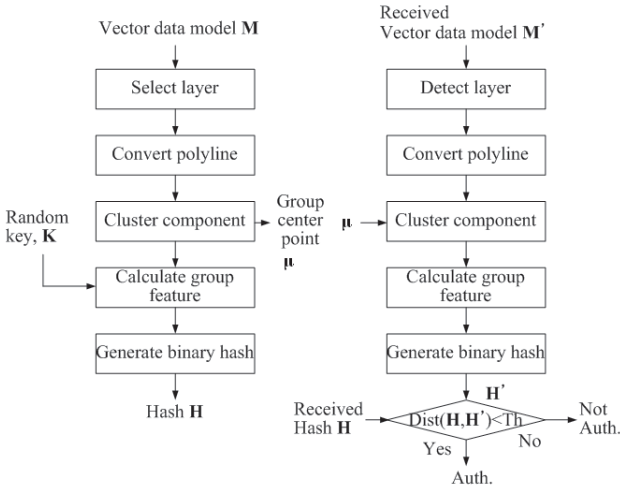


Fig. 1 The process of hash generation and hash authentication in vector data model.

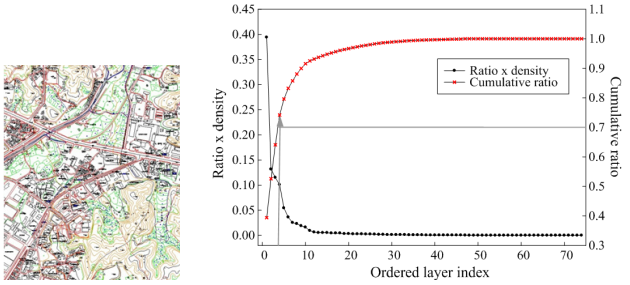


Fig. 2 (a) 1:5,000 scaled digital map and (b) the ratio of the accumulated polyline number in layers that are arranged in descending order of $\gamma\rho$.

3. Proposed Vector Data Hashing

Generally, lines, polylines, and polygons among primitive entities occupy most of the vector data model. We convert the lines, polylines, and polygons to polylines that are available to the first and second curvatures and select target layers with the highest number of converted polylines. We then extract the feature values through the polyline curvatures in each layer and generate the binary hash. The overall process of the proposed hashing scheme consists of layer selection, polyline refinement, polyline clustering, feature extraction, and hash generation, as shown in Fig. 1.

In this paper, we consider lines, polylines, and polygons to be polylines for the sake of simple notation, although they are definitely different. The main notations used in this paper are as follows. Let i be the index of layer and j be the index of a polyline in any layer. A vector data model $\mathbf{M}$ consists of N layers; $\mathbf{M} = \{\mathbf{L}_i | i \in [1, N]\}$. A layer $\mathbf{L}_i$ consists of $N(\mathbf{L}_i)$ polylines; $\mathbf{L}_i = \{\mathbf{p}_{ij} | j \in [1, N(\mathbf{L}_i)]\}$. A polyline $\mathbf{p}_{ij}$ has N_{ij} vertices; $\mathbf{p}_{ij} = \{\mathbf{p}_{ij,k} | k \in [1, N_{ij}]\}$, which is a jth polyline in the ith layer. The feature $\mathbf{F}$ is the matrix of $N_H \times N_H$ real numbers; $\mathbf{F} = \{f_{n_1,n_2} | n_1, n_2 \in [1, N_H]\}$. The hash $\mathbf{H}$ is the matrix of $N_H \times N_H$ bits; $\mathbf{H} = \{h_{n_1,n_2} | n_1, n_2 \in [1, N_H]\}$.

3.1 Layer Selection

Each layer of a vector data model has its own geometric property according to the kind of model. For example, the road layer in a GIS digital map has a number of polylines of the straight type and polygons of the rectangular type. The watch layer in design drawings has a number of arcs and curve-type polylines. The hash must be generated from the primary layers in a vector data model for robust hashing. Conventional watermarking techniques for CAD drawings [3] selected primary layers based on the density of geometric primitives. Our algorithm selects the primary layers using the multiple factor $\gamma_i\rho_i$ where γ_i is the ratio of the polyline number to the total number of other primitives and ρ_i is the polyline density on each layer $\mathbf{L}_i$.

Let $N(\mathbf{L}_k)$ be the number of polylines in a layer $\mathbf{L}_k$ and η_i be the ratio of the accumulated polyline number, $\sum_{k=1}^{i} N(\mathbf{L}_k)$, in i layers ($i \le N$) to the total polyline number, $\sum_{k=1}^{N} N(\mathbf{L}_k)$, in a model. Then, we arrange all layers in descending order of $\gamma_i\rho_i$, and select layers from the highest to the lowest layer while η_i is above Th_η. Therefore, the selected layer $\mathbf{L}^*$ is defined by

$$\mathbf{L}^* = \{\mathbf{L}_1, \cdots, \mathbf{L}_i, \cdots, \mathbf{L}_{N_L}\}, \quad \forall\ \eta_i > Th_\eta \text{ and } \gamma_{\mathrm{i}}\rho_{\mathrm{i}} > \gamma_{\mathrm{i+1}}\rho_{\mathrm{i+1}},\ \mathrm{i} \in [1, \mathrm{N_L}] \tag{3}$$

This means that the number of polylines in $\mathbf{L}^*$ is more than $Th_\eta \times 100[\%]$ of the number of total polylines. If Th_η is small, the number of the selected layers and hash length increase; however, the robustness decreases because insignificant layers are selected. In contrast, if Th_η is large, a few of the main layers are selected and the robustness increases; however, the hash length decreases and reduces security. Therefore, we determine Th_η to be 0.7 so that the number N_L of selected layers is above 2.

Figure 2 shows a 1:5,000 scale digital map and the η ratios in layers arranged by descending order of $\gamma_i\rho_i$. This figure illustrates that polylines are distributed intensively on a few of the layers; in particular, 4 layers have over 70% of the polylines of the map. As in the results above, we confirmed that polylines are distributed largely on the main layers in most models.

3.2 Polyline Refinement and Clustering

The foremost three vertices of the polyline must be used for the initialization of the second curvature in Eq. (2). We refine the short polylines with three or less vertices to be thrice differentiable polylines by the linkage of nearest polylines. Let the short polyline be $\mathbf{p}^s_{ij} = \{\mathbf{v}_{ij,1}, \cdots, \mathbf{v}_{ij,N_{ij}}\}, N_{ij} < 4$. The refinement process links $\mathbf{p}^s_{ij}$ virtually to short polylines that are the nearest to the start vertex $\mathbf{v}_{ij,1}$ and the end vertex $\mathbf{v}_{ij,N_{ij}}$ within the range of the average length of $\mathbf{p}^s_{ij}$. If $\mathbf{p}^s_{ij}$ has no polylines within the range of the average polyline length, $\mathbf{p}^s_{ij}$ will not be selected to the curvature. Figure 3 illustrates an example that a current line $\mathbf{p}^s_{ij}$ is refined by the sequential

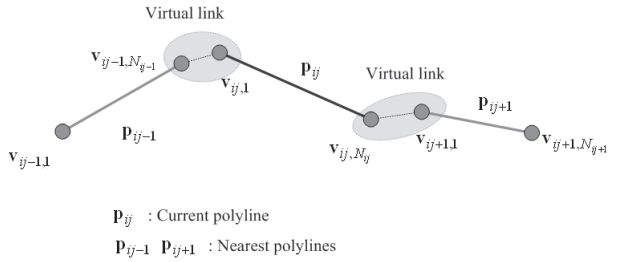


Fig. 3 An example of polyline refinement of $\mathbf{p}_{ij}^{s}$ to link between the nearest polylines and $\mathbf{p}_{ij}^{s}$.

linkage of left and right nearest polylines, $\mathbf{p}_{ij-1}^{s}$ and $\mathbf{p}_{ij+1}^{s}$.

The layers and primitive entities in the vector data are searched by their indices. However, the index of each layer or entity is variable. Therefore, layers and primitive entities can be rearranged by the random permutation of indices. If the relation between a hash bit and a polyline is one-to-one, a number of hash bits will be produced, although the hash cannot be extracted without knowledge of the polylines indices. Thus, the hash can be easily broken by the rearrangement of layers or polylines without degrading the quality. Our algorithm clusters polylines using their curve energies and generates a hash bit for each cluster. As a result, the hash is extracted directly in the rearranged vector data and is robust against geometric attacks, although its length is reduced.

In the clustering process, we first calculate the bending energies [29] of all the polylines in the selected layer $\mathbf{L}_i$. For a smooth curve $\Upsilon(t)$, defined in some parameter interval $[a, b]$, the linearized bending energy (cubic spline energy) is defined by $E(\Upsilon) = \int_a^b \|\Upsilon^{(2)}(t)\|^2 dt$. Here, $\Upsilon^{(2)}$ denotes the second derivative vector of the curve $\Upsilon(t)$ with respect to the curve parameter t. A polyline $\mathbf{p}_{ij}$ as a discrete curve possesses a discrete linearized bending energy $E[\mathbf{p}_{ij}]$, which is called "bending energy" for short.

$$E[\mathbf{p}_{ij}] = \sum_{k=2}^{N_{ij-1}} \|\Delta^2 \mathbf{v}_{ij,k}\|^2, \tag{4}$$
$$\Delta^2 \mathbf{v}_{ij,k} = \mathbf{v}_{ij,k-1} - 2\mathbf{v}_{ij,k} + \mathbf{v}_{ij,k-1}$$

We then cluster all polylines into N_H groups, which is the length of hash, using the normalized bending energy $E[\mathbf{p}_{ij}]$

$$\bar{E}[\mathbf{p}_{ij}] = \frac{E[\mathbf{p}_{ij}]}{\sum_{j=1}^{N(\mathbf{L}_i)} E[\mathbf{p}_{ij}]} \tag{5}$$

We denote the set of N_H groups as $\mathbf{G}_i = \{\boldsymbol{G}_{i,n} | n \in [1, N_H]\}$. Polylines can be clustered by many algorithms such as k-means/k-means++ clustering [30], fuzzy c-means clustering [31], expectation-maximization (EM) clustering [32], [33], quality-threshold (QT) clustering [34], and spectral clustering. Our algorithm uses Gaussian mixture model (GMM)-based EM clustering because of its sensitivity to initial values, which enable it to improve the security of hash.

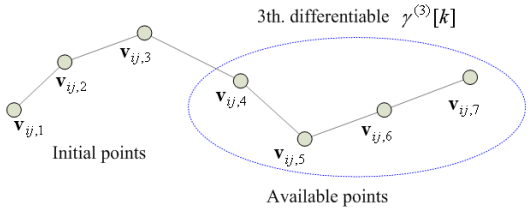


Fig. 4 The initial 3 vertices and the vertices that are available to Frenet vectors in a polyline $\mathbf{p}_{ij} = (\mathbf{v}_{ij,1}, \cdots, \mathbf{v}_{ij,N_{ij}})$.

Let the distribution of $\bar{E}[\mathbf{p}_{ij}]$ be the sum of Gaussian distributions weighted by $\omega_n; \{\phi(\mu_n, \sum_n)\}_{n=1}^{N_H}$, where μ_n is the mean and $\sum_n$ is the covariance. A weight ω_n has the property of $\omega_n > 0$ and $\sum_{n=1}^{N_H} \omega_n = 1$. Let θ_n be parameters of weight, mean, and covariance of nth group; $\theta_n = (\omega_n, \mu_n, \sum_n)$. We randomly select N_H initial parameters $\{\theta_n^{(0)}\}_{n\in[1,N_H]}$ in the selected layer $\mathbf{L}_i$ and obtain N_H GMM-based groups by finding parameters $\{\theta_n\}_{n\in[1,N_H]}$ that maximize the log-likelihood of the normalized bending energy.

$$\boldsymbol{G}_{i,n} = \{\mathbf{p}_{ij} : \Pr(\bar{E}[\mathbf{p}_{ij}]\theta_n) > \Pr(\bar{E}[\mathbf{p}_{ij}]\theta_{n'}), \quad \forall n' \neq n \in [1, N_H]\} \tag{6}$$

This set of parameters $\Theta = \{\theta_n\}_{n\in[1,N_H]}$ is stored to extract the hash. All polylines in a layer $\mathbf{L}_i$ are clustered to N_H groups; $\{\boldsymbol{G}_{i,n}\}_{n\in[1,N_H]}$.

3.3 Hash Feature Value

3.3.1 Polyline Curvature

The average of the first and second curvatures, $\chi_1(\mathbf{p}_{ij})$ and $\chi_2(\mathbf{p}_{ij})$, of polylines in the selected layer $\mathbf{L}_i$ are calculated from the following equation derived from Eq. (2).

$$\chi_1(\mathbf{p}_{ij}) = \sum_{k=3}^{N_{ij}} \frac{\chi_1[k]}{N_{ij}-3}, \chi_2(\mathbf{p}_{ij}) = \sum_{k=3}^{N_{ij}} \frac{\chi_2[k]}{N_{ij}-3} \tag{7}$$

As shown in Fig. 4, we use the three vertices, $\mathbf{v}_{ij,1}, \mathbf{v}_{ij,2}, \mathbf{v}_{ij,3}$, for the initial points because the third Frenet vector $\mathbf{e}_3[k]$ is to be thrice differentiable. The second and third Frenet vectors, $\mathbf{e}_2[k]$ and $\mathbf{e}_3[k]$ at a vertex $\mathbf{v}_{ij,k}(k \in [3, N_{ij}])$ can be obtained as follows;

$$\mathbf{e}_2[k] = \frac{\bar{\mathbf{e}}_2[k]}{\|\bar{\mathbf{e}}_2[k]\|}, \mathbf{e}_3[k] = \frac{\bar{\mathbf{e}}_3[k]}{\|\bar{\mathbf{e}}_3[k]\|} \tag{8}$$

where $\bar{\mathbf{e}}_2[k] = \Upsilon^{(2)}[k] - < \Upsilon^{(1)}[k], \mathbf{e}_1[k] > \mathbf{e}_1[k]$, $\bar{\mathbf{e}}_3[k] = \Upsilon^{(3)}[k] - \sum_{j=1}^{2} < \Upsilon^{(j)}[k], \mathbf{e}_j[k] > \mathbf{e}_j[k]$.

The first derivatives of the first and second Frenet vectors, $\mathbf{e}_1'[k]$ and $\mathbf{e}_2'[k]$, can be calculated by Frenet-Serret formula [22],

$$\begin{bmatrix} \mathbf{e}_1'[k] \\ \mathbf{e}_2'[k] \end{bmatrix} = \|\Upsilon^{(1)}[k]\| \begin{bmatrix} 0 & \chi_1[k] \\ -\chi_1[k] & 0 \end{bmatrix} \begin{bmatrix} \mathbf{e}_1[k] \\ \mathbf{e}_2[k] \end{bmatrix} \tag{9}$$

As described above, we calculate pairs of the first and second average curvatures, $(\bar{\chi}_1(\mathbf{p}_{ij}), \bar{\chi}_2(\mathbf{p}_{ij}))$, of all polylines in each group.

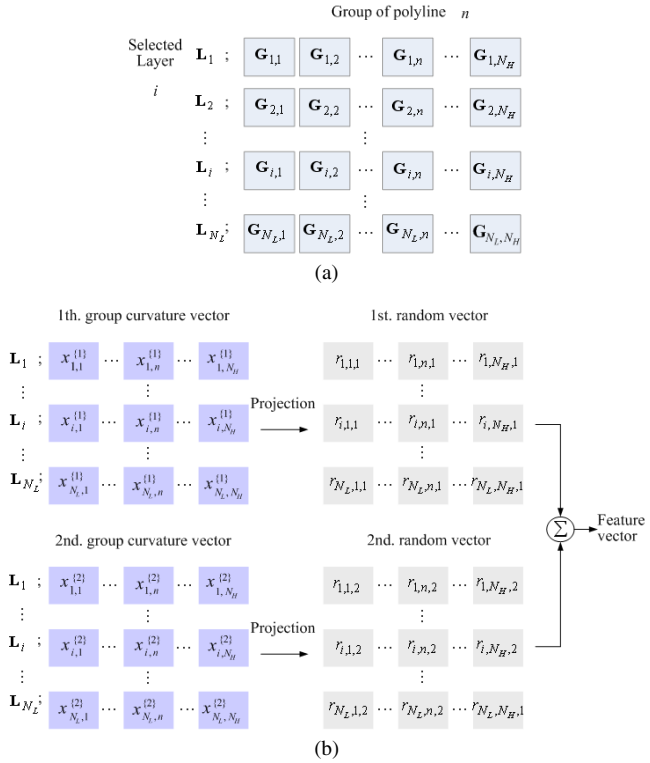


Fig. 5 (a) Polylines group in each layer and (b) extraction of the feature vector using two curvature vectors and two random vectors.

3.3.2 Feature Matrix of Group

We define the values for representing a group as two averages of the first and second curvatures, which are called the first group value and the second group value. Thus, the first group value $x_{i,n}^{\{1\}}$ and the second group value $x_{i,n}^{\{2\}}$ in a group $\boldsymbol{G}_{i,n}$ are defined as follows.

$$x_{i,n}^{\{1\}} = \sum_{j=1}^{N_{i,n}} \frac{\bar{\chi}_1(\mathbf{p}_{ij})}{N_{i,n}}, x_{i,n}^{\{2\}} = \sum_{j=1}^{N_{i,n}} \frac{\bar{\chi}_2(\mathbf{p}_{ij})}{N_{i,n}}, \quad \text{for } \mathbf{p}_{ij} \in \boldsymbol{G}_{i,n} \tag{10}$$

where $N_{i,n}$ is the number of polylines in a group $\boldsymbol{G}_{i,n}$.

Since the number of selected layers is N_L and all polylines in a layer are clustered to N_H groups, the number of groups of all selected layers is $N_L \times N_H$, as shown in Fig. 5 (a). Thus, the first and second group values in all selected layers can be written as matrices $\mathbf{X}^{\{1\}}, \mathbf{X}^{\{2\}}$ of size $N_L \times N_H$, as shown in Fig. 5 (b).

$$\mathbf{X}^{\{1\}} = \{x_{i,n}^{\{1\}} | i \in [1, N_L], n \in [1, N_H]\}, \tag{11}$$
$$\mathbf{X}^{\{2\}} = \{x_{i,n}^{\{2\}} | i \in [1, N_L], n \in [1, N_H]\},$$

We generate two Gaussian random matrices, $\mathbf{R}_1$, $\mathbf{R}_2$ with mean m_R and variance σ_R^2 of size $N_L \times N_H$, and then calculate the group feature matrix $\mathbf{F}$ of size $N_H \times N_H$, which is defined by the products of $[\mathbf{X}^{\{1\}}]^{\mathrm{T}}$ and $\mathbf{R}_1$ and of $[\mathbf{X}^{\{2\}}]^{\mathrm{T}}$ and $\mathbf{R}_2$.

$$\mathbf{F} = [\mathbf{X}^{\{1\}}]^{\mathrm{T}}\mathbf{R}_1 + \alpha[\mathbf{X}^{\{2\}}]^{\mathrm{T}}\mathbf{R}_2 \tag{12}$$

The feature value is written as

$$f_{n_1,n_2} = \sum_{i=1}^{N_L} x_{i,n_1}^{\{1\}} r_{1,i,n_2} + \alpha \sum_{i=1}^{N_L} x_{i,n_1}^{\{2\}} r_{2,i,n_2} \quad \forall n_1, n_2 \in [1, N_H] \tag{13}$$

$[\mathbf{X}]^{\mathrm{T}}$ is the transposed matrix of $\mathbf{X}$. The intensity α equalizes the dynamic range of $x_{i,n}^{\{1\}}$ and $x_{i,n}^{\{2\}}$.

$$\alpha = max\{x_{i,n}^{\{1\}}\}/max\{x_{i,n}^{\{2\}}\} \tag{14}$$

3.4 Hash Generation

The binary hash $\mathbf{H}$ is generated from the binarization of the group feature value $\mathbf{F}$.

$$h_{n_1,n_2} = \begin{cases} 0, & f_{n_1,n_2} < Th_h \\ 1, & f_{n_1,n_2} \geq Th_h \end{cases} \tag{15}$$

The threshold Th_h is determined for the binary hash as a Bernoulli distribution with $\Pr[h_{n_1,n_2} = 1] \approx \Pr[h_{n_1,n_2} = 0] \approx 0.5$, as follows.

1) Select an initial threshold $Th_h^{(0)}$ randomly in the dynamic range of f_{n_1,n_2}.

2) Classify feature values to two groups $\mathbf{G}_1$, $\mathbf{G}_2$ using $Th_h^{(t)}(t \geq 0)$, $\mathbf{G}_1 = \{f_{n_1,n_2} | f_{n_1,n_2} \geq Th_h^{(t)}\}$, $\mathbf{G}_2 = \{f_{n_1,n_2} | f_{n_1,n_2} < Th_h^{(t)}\}$.

3) Update $Th_h^t = (m_1 + m_2)/2$ where m_1, m_2 are average feature values in each group.

4) Perform iteratively until $Th_h^{(t)} \approx Th_h^{(t-1)}$. If $Th_h^{(t)}$ is converged, Th_h is set to $Th_h^{(t)}$.

3.5 Hash Authentication

As shown in Fig. 1 (b), the proposed hashing algorithm extracts the hash $\mathbf{H}'$ in the received vector data model $\mathbf{M}'$ from a process similar to that of hash generation, and determines the authentication of $\mathbf{M}'$ using the normalized Hamming distance $d(\mathbf{H}, \mathbf{H}')$

$$d(\mathbf{H}, \mathbf{H}') = \frac{1}{N_H^2} \sum_{n_1}^{N_H} \sum_{n_2}^{N_H} |h_{n_1,n_2} - h'_{n_1,n_2}| \tag{16}$$

$$\mathbf{M}' \rightarrow \begin{cases} Auth. & d(\mathbf{H}, \mathbf{H}') < \epsilon \\ NotAuth. & d(\mathbf{H}, \mathbf{H}') \geq \epsilon \end{cases} \tag{17}$$

$d(\mathbf{H}, \mathbf{H}')$ converges to 0 if the two models are perceptually identical and to 0.5 if they are perceptually different.

We performed a test of the hypothesis based on the receiver operating characteristic (ROC) for determining the authentication threshold ϵ. The test extracted 5,000 hashes in 50 models using 100 keys, and calculated $d(\mathbf{H}, \mathbf{H}')$ between the original and attacked hashes and measured the true positive (TP) and false positive (FP) rates while varying ϵ from 0 to 0.5 in steps of 0.05. TP is the probability $p_{TP} = \Pr[d(\mathbf{H}, \mathbf{H}') < \epsilon]$ that the hash $\mathbf{H}'$ of attacked model $\mathbf{M}'$ is authenticated to the original model $\mathbf{M}$, and FP is the

Table 1 The number of layers and polylines, differential entropy, and processing time of typical vector data models.

Test model		Total number		Selected number		Polyline ratio [%] (N/T*100)	Differential entropy			Processing Time [ms]
		Layer	Polyline (T)	Layer	Polyline (N)		Average	Minimum	Maximum	
Drawing	Block and Tables	16	76	2	55	72.4	9.03	6.85	12.39	646.78
	Db	29	1,737	2	1,351	77.8	10.61	8.52	14.45	2,293.14
	Hummer Elevation	30	1,492	1	1,492	100.0	11.02	8.91	14.60	2,354.00
	Stadium Elevation	52	5,442	2	3,916	71.9	12.02	9.81	15.17	4,058.11
	Tablet	5	1,458	2	1,201	82.4	11.06	8.51	14.23	1,473.12
	Taisei Detail Plan	37	2,288	1	2,288	100.0	11.19	9.24	14.84	2,922.14
Map	1:1,000	72	633	2	481	75.9	10.77	8.28	13.71	1,218.56
	1:5,000	74	2,489	3	1,851	74.4	11.52	9.57	14.89	3,017.49
	1:25,000	116	18,911	6	16,361	86.5	12.46	10.02	16.90	9,811.93
	1:250,000	29	2,001	4	1,530	76.5	11.33	8.96	14.61	2,688.82

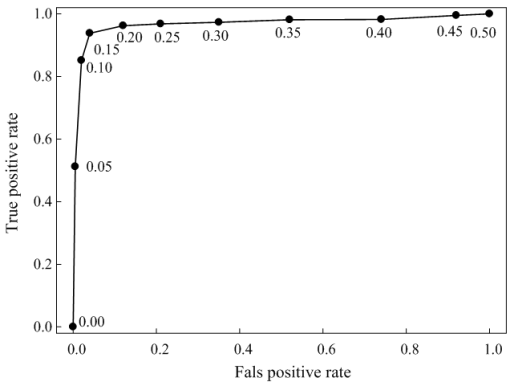


Fig. 6 ROC plot on the authentication threshold ϵ.

probability $p_{FP} = \Pr[d(\mathbf{H}, \mathbf{H}') < \epsilon]$ that the former is authenticated to a different model $\mathbf{M}_k$. As shown in Fig. 6, the ROC plot illustrates that TP achieves a maximum of 0.95 and FP reaches a minimum of 0.04 when ϵ is 0.15. Therefore, we determine ϵ to be 0.15.

4. Experimental Results

Our experiment used 50 vector data models provided by the AutoCAD software [35] in autodesk and NGII [36] for evaluating the robustness, uniqueness, and security of our algorithm. We set the number of polyline groups to N_H=15 and the bit length of the hash to $N_H \times N_H$=225, although the numbers of layers and polylines are different in each model. Table 1 shows the number of layers and polylines of a typical vector data model. From this table, we see that most of polylines are distributed in a few of layers in the design drawings and digital maps and the selected layers contain more than 70% of the total polylines.

4.1 Robustness Evaluation

We attacked test vector data models using various editing functions in AutoCAD S/W to evaluate robustness using the false detection probability as the evaluation measure.

$$p_{FD}(\delta, \mathbf{H}, \mathbf{H}') = 1 - \Pr[d(\mathbf{H}, \mathbf{H}'), \epsilon|\delta] \quad (18)$$

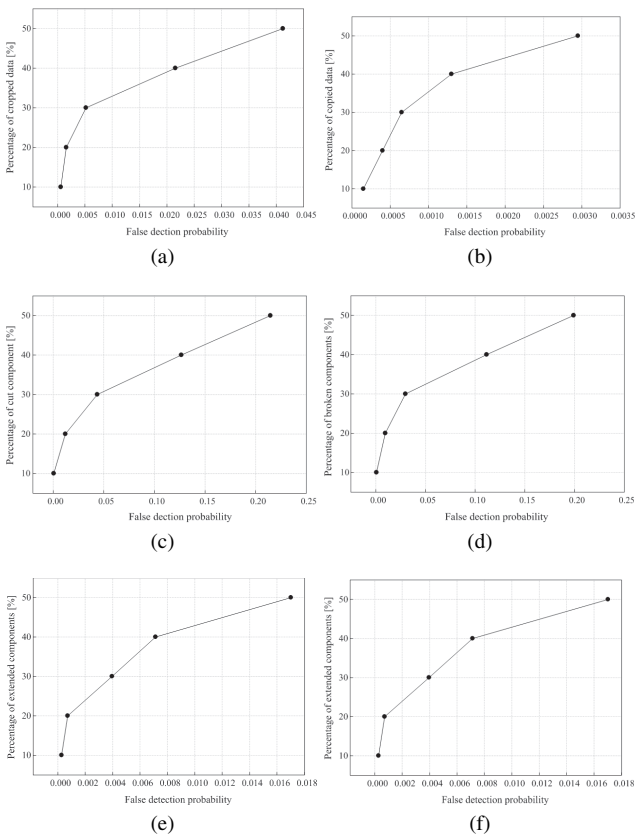


Fig. 7 False detection probabilities in attacks of (a) cropping, (b) copying, (c) trimming, (d) breaking, (e) extending, and (f) polyline simplification.

p_{FD} is the probability that $d(\mathbf{H}, \mathbf{H}')$ of an original hash $\mathbf{H}$ and an attacked hash $\mathbf{H}'$ at any attack intensity δ is above the threshold ϵ. We calculated p_{FD} by iteratively performing random key generation and hash extraction until $d(\mathbf{H}, \mathbf{H}') > \epsilon$,($\epsilon = 0.15$) at the attack intensity δ. The experimental results of the robustness evaluation are shown in Fig. 7.

4.1.1 Similarity Transform and Rearrangement

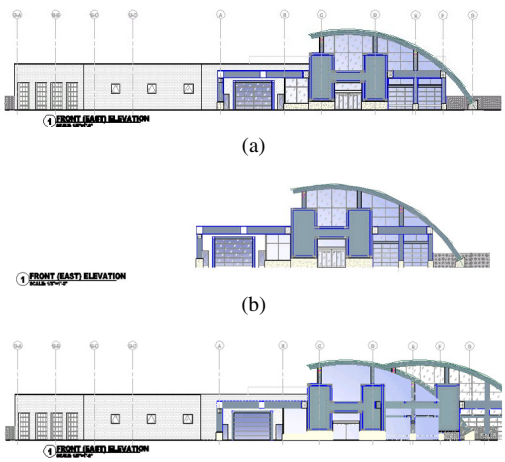
The polyline curvature of the feature value is invariant to

rotation and translation but is linearly variable to scaling. Our algorithm estimates the scaling factor s by comparing the bending energy E of the original model and the bending energy E' of the scaled model, and then rescales the scaled model so that E' is $1/s$. From this rescaling process, we confirmed that the hash could be detected without error in an arbitrary scaled model in which p_{FD} is 0. Layers and geometric primitives can be rearranged easily by permutation of their own indices while the quality of the model is not degraded at all. However, our algorithm is unaffected by the rearrangement because of the selection of a target layer based on the polyline density and polyline clustering by the bending energy. Therefore, p_{FD} is 0 in the rearranged model.

4.1.2 Object Cropping and Copying

The geometric primitives of a vector data model can be cropped or copied without being related to each other. We cropped and copied about 10–50% of the total primitives in a model. Figure 8 shows the front of the first floor in the Hummer Elevation drawing and in a drawing in which 30% of the primitives were cropped or copied. The cropped drawing shows that the main layers are left intact but other parts are cropped. The copied drawing shows that the right portions of the main layers are copied repeatedly.

Figure 7 (a) and 7 (b) illustrate the results of cropping and copying. The p_{FD} is very low from 6×10^{-4} to 4.2×10^{-2} up to 50% cropping of the data and 1.5×10^{-4} to 2.9×10^{-3} up to 50% of copied data, which is lower than the p_{FD} of cropping. The first and second group values in Eq. (10) have not been affected by a few cropped polylines because they are computed by the average of the curve energies of the polylines. However, if a number of polylines is cropped, these group values will be strongly affected or not exist. Our hashes are generated by the feature values in Eq. (13), which are the combination of the group values of all the groups. Therefore, though any groups will be strongly affected or not exist as a result of the cropping, our hashes can be preserved with the values of the remaining groups. Furthermore, copied polylines do not affect the group values strongly because they are repeated. These results verified that the hash has robustness to cropping and copying.



(a)

(b)

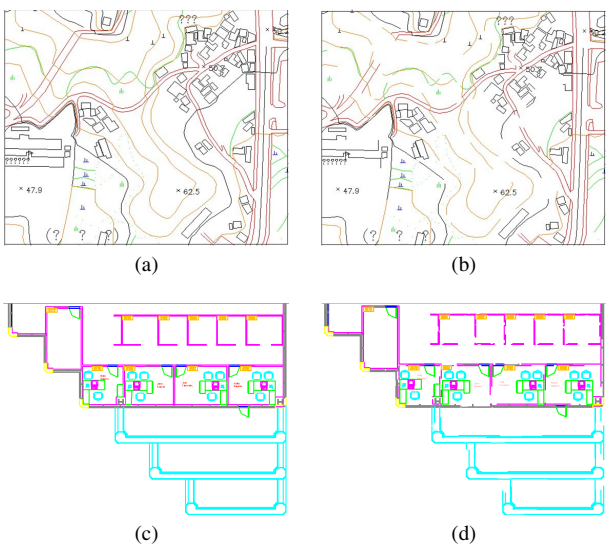
(c)

Fig. 8 The front of first floor of (a) Hummer Elevation drawing, (b) 30% data cropped drawing, (c) 30% data copied drawing.

4.1.3 Trimming and Breaking

The trimming process trims the corners of polylines or continuous polylines or polygons on any point. The breaking splits a polyline into two sub-polylines or removes the end of one side. It removes the part between the user-defined points in a polyline. Our experiment selected randomly 10–50% of the polylines and trimmed any line or broken on these polylines. Figure 9 (a) shows an area of a 1:5,000 scale map, and Fig. 9 (b) shows that of a map for which 50% of the polylines were trimmed. Figure 9 (a) shows the bottom of the original Db drawing, and Fig. 9 (b) shows that of a Db drawing in which 30% of the polylines were broken.

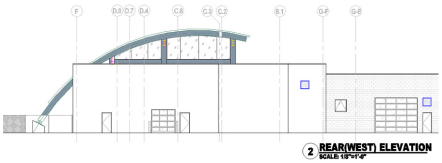
The result of polyline trimming and breaking, as shown in Fig. 7 (c) and 7 (d), reveal that p_{FD} is very low from 6.0×10^{-4} to 4.3×10^{-2} with 10–30% of the polylines trimmed and 7.5×10^{-4} to 2.9×10^{-2} with 10-30% of the polylines broken. But it is somewhat high from 1.1×10^{-1} to 2.1×10^{-1} with 40–50% trimmed and broken. Trimmed or broken polylines can be allocated to other groups because of the change in their normalized bending energy, as in Eq. (5). In addition, their first and second curvatures can be changed slightly or not. These affect the first and second group values of both the origin group and the reallocated group and then affect slightly the feature values, which are a combination of the group values, according to the number of trimmed or broken polylines. From experimental results, we see that our hashing has a p_{FD} of less than 4.3% with up to 30% of the polylines trimmed and broken, and has good robustness



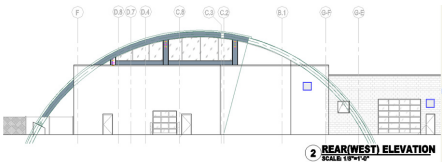
(a) (b)

(c) (d)

Fig. 9 An area of 1:5000 scaled (a) original map and (b) map with 50% trimmed polylines and the bottom of (c) original Db drawing and (d) Db drawing with 30% broken polylines.



(a)



(b)

Fig. 10 The rear of (a) original Hummer Elevation drawing and (b) drawing with 20% extended polylines.

up to 30% trimming and breaking.

4.1.4 Extending

The extending process extends any polyline for being connected to the corner of a user-defined polyline. In an extending experiment, we selected 10–50% of the polylines as candidate and reference polylines, and extended the former up to the corners of the latter. Figure 10 (a) shows the rear of the Hummer Elevation drawing, and Fig. 10 (b) shows that of a drawing for which 20% polylines were extended.

The result of the extending, as shown in Fig. 7 (e), reveals that the p_{FD} is very low from 1.4×10^{-4} to 8.5×10^{-3} with 10–50% of the polylines extended. Since extending increases the length of the polyline while preserving the shape, it does not affect the first and second curvatures, which depend not on the length but on the curve. Accordingly, we verified that our hash is robust to extending up to 50%.

4.1.5 Simplification

Simplification, which is used mainly for data compression, reduces the vertices that preserve the shape of an object. In an experiment, we reduced 10–50% of the vertices in all polylines using the subsample. The result of the simplification, as shown in Fig. 7 (f), reveals that the p_{FD} is low from 1.6×10^{-3} to 2.5×10^{-2} with 10–50% of the polylines simplified. The simplification causes the curve to lack smoothness, while preserving the shape of the polylines. It somewhat affects the first and second curvatures but affects less the normalized bending energy and the group values. Accordingly, we verified that our hash is robust at up to 50% of simplification.

From the above results, we see that our algorithm is robust up to the test range of data cropping and copying, extending, and simplifying, and robust up to a specific range of trimming and breaking, which are the most common functions in vector data editing tools. In addition, our algorithm is not affected by similarity transformation of rotation, scaling, translation (RST), and data rearrangement.

Table 2 Results of uniqueness evaluation.

Category	Model Uniq.	Key Uniq.
$\Pr[d(\mathbf{H}_1,\mathbf{H}_2) \geq 0.5-e]$	0.956987	0.98
$\Pr[e \leq d(\mathbf{H}_1,\mathbf{H}_2) < 0.5-e]$	0.043012	0.02
$\Pr[d(\mathbf{H}_1,\mathbf{H}_2) < e]$	0.000001	0.00

4.2 Uniqueness Evaluation

The uniqueness of a hash can be evaluated by models and keys. Model uniqueness refers to the fact that two hashes $\mathbf{H}_1, \mathbf{H}_2$ generated in different models $\mathbf{M}_1, \mathbf{M}_2$ with the same key $\mathbf{K}$ are statistically very different.

$$\Pr[\mathbf{H}_1 \neq \mathbf{H}_2] \approx 1 \quad (19)$$
$$\text{where } \mathbf{H}_1 = hash(\mathbf{M}_1, \mathbf{K}), \mathbf{H}_2 = hash(\mathbf{M}_2, \mathbf{K})$$

Key uniqueness refers to the fact that two hashes $\mathbf{H}_1$, $\mathbf{H}_2$ generated in different keys $\mathbf{K}_1$, $\mathbf{K}_2$ with the same model $\mathbf{M}$ are statistically very different.

$$\Pr[\mathbf{H}_1 \neq \mathbf{H}_2] \approx 1 \quad (20)$$
$$\text{where } \mathbf{H}_1 = hash(\mathbf{M}, \mathbf{K}_1), \mathbf{H}_2 = hash(\mathbf{M}, \mathbf{K}_2)$$

Thus, the normalized Hamming distance $d(\mathbf{H}_1, \mathbf{H}_2)$ of two hashes, in model and key uniqueness experiments, must approach 0.5.

We evaluated the normalized Hamming distances of hashes generated in 50 models with the same key, and those of hashes generated in a model with 1,000 keys. Table 2 shows the results of model and key uniqueness experiments, which are classified into three categories. $\Pr[d(\mathbf{H}_1, \mathbf{H}_2) \geq 0.5-\epsilon]$ indicates that the two hashes are statistically very different, and $\Pr[d(\mathbf{H}_1, \mathbf{H}_2) < \epsilon]$ indicates that the two hashes are statistically very similar. In the results for model uniqueness, $\Pr[d(\mathbf{H}_1, \mathbf{H}_2) \geq 0.5 - \epsilon]$ is 0.95 and $\Pr[d(\mathbf{H}_1, \mathbf{H}_2) < \epsilon]$ is 10^{-6}. Thus, there is a 95% probability that the hashes in different models using the same key are different from each other. In the results for key uniqueness, $\Pr[d(\mathbf{H}_1, \mathbf{H}_2) \geq 0.5 - \epsilon]$ is 0.98 and $\Pr[d(\mathbf{H}_1, \mathbf{H}_2) < \epsilon]$ is 0. Thus, there is a 98% probability that the hashes in different keys using the same model are different from each other.

4.3 Security Analysis

Swaminathan [16] presented an analysis and evaluation of hash security based on differential entropy. Following this evaluation, we model the probability density function $p(f)$ of the group feature value and calculate the differential entropy $H(f) = -\int_F p(f)log(f)df$. f_{n_1,n_2} can be rewritten as the sum of two variables, $A = \sum_{i=1}^{N_L} x_{i,n_1}^{\{1\}} r_{1,i,n_2}$ and $B = \alpha \sum_{i=1}^{N_L} x_{i,n_1}^{\{2\}} r_{2,i,n_2}$, in Eq. (13). Given that two random values r_1, r_2 for the first and second group, $x_{i,n}^{\{1\}}$ and $x_{i,n}^{\{2\}}$, are (m_1, σ_1^2) and (m_2, σ_2^2), the two variables A, B are normally distributed, $(m_A, \sigma_A^2) = (\sum_{i=1}^{N_L} x_{i,n_1}^{\{1\}} m_1, \sum_{i=1}^{N_L} (x_{i,n_1}^{\{1\}})^2 \sigma_1^2)$. Therefore, $p(f)$ is normally distributed with a mean m_f and a variance σ_f^2

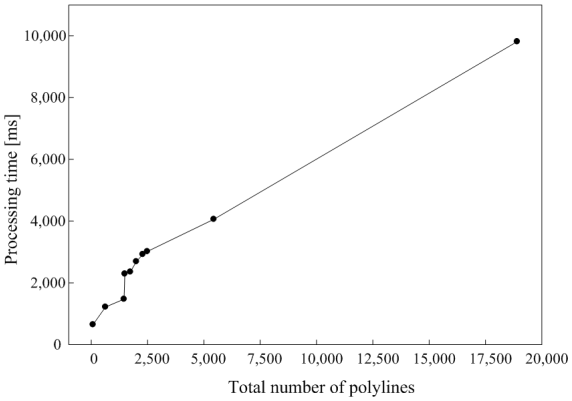


Fig. 11 Processing time of test models.

$$m_f = \sum_{i=1}^{N_L} x_{i,n_1}^{\{1\}} m_1 + \sum_{i=1}^{N_L} x_{i,n_1}^{\{2\}} m_2 \tag{21}$$

$$\sigma_f^2 = \sum_{i=1}^{N_L} (x_{i,n_1}^{\{1\}})^2 \sigma_1^2 + \sum_{i=1}^{N_L} (\alpha x_{i,n_1}^{\{2\}})^2 \sigma_2^2 \tag{22}$$

The differential entropy $H(f)$ of $p(f)$ is

$$\begin{aligned} H(f) &= \frac{1}{2}\log(2\pi e\sigma_f^2) \\ &= \frac{1}{2}\log(2\pi e(\sum_{i=1}^{N_L}(x_{i,n_1}^{\{1\}})^2\sigma_1^2 + \sum_{i=1}^{N_L}(\alpha x_{i,n_1}^{\{2\}})^2\sigma_2^2)). \end{aligned} \tag{23}$$

If the two random values are the same as (m_R, σ_R^2), $H(f)$ will be $H(f) = \frac{1}{2}\log(2\pi e\sigma_R^2)(\sum_{i=1}^{N_L}(x_{i,n_1}^{\{1\}})^2 + \sum_{i=1}^{N_L}(\alpha x_{i,n_1}^{\{2\}})^2)$. We used the normal distribution of (1,1) for the two random variables and calculated the average, maximum, and minimum $H(f_{n_1,n_2})$ of the group feature values, as shown in Table 1. The differential entropy varies according to the number of layers and polylines as well as the curvatures; it ranges from 9.03 to 12.46 on average.

4.4 Processing Time

To evaluate the algorithm complexity, we measured the processing time experimentally, instead of the operation count. In our experiments, we used a computer whose system was Intel Core Duo CPU, 1.2 GHz, 1 GB RAM. Our algorithm takes time to process 1) the layer selection for all layers, and 2) the refinement/clustering and the hash generation by feature matrix for selected layers. The latter process takes a much longer time than the former. The processing time is affected by the number of polylines in each layer and the number of vertices in each polyline, which are different in vector models. The processing times [ms] for test models are shown in Table 1, on the right. It is from 0.65 to 9.81 s in typical test models. Figure 11 illustrates the processing time for the number of polylines (N) in the selected layers. These results verify that the processing time is close to $O(N\log_2(N))$ complexity and our algorithm is somewhat effective in terms of the processing time.

5. Conclusions

We presented a vector data-hashing algorithm using polyline curvature for authentication and copyright protection of a vector data model. In the proposed hashing scheme, we clustered polylines into selected layers according to their curve energies and calculated group values for the average curvatures of the first and second type. We then obtained the feature values through a combination of the group and random values, and generated the final hash from the binarization of the feature value. Experimental results verified that the proposed hashing algorithm is robust against RST, cropping/copying, breaking, trimming, extending, and simplifying, The algorithm also has security and uniqueness due to random keys. Therefore, the proposed hashing scheme can be very useful in security techniques for CAD drawings and GIS digital maps.

In future work, we intend to adapt the scheme to various geometric primitives, such as arc and circle, which are the main primitives in Watch design. We also intend to adapt it to additional information besides polylines and polygons such as text, and to model the mutual relationships among robustness, security, and uniqueness.

References

[1] K. Chang, Introduction to Geographic Information System, 4th ed., McGraw-Hill, 2007.
[2] G. Farin, J. Hoschek, and M.-S. Kim, Handbook of Computer Aided Geometric Design, Elsevier science, 2002.
[3] S.H. Lee and K.R. Kwon, "CAD drawing watermarking scheme," Digit. Signal Process., vol.20, Issue 5, pp.1379–1399, Sept. 2010.
[4] F. Penga, R.-S. Guoa, C.-T. Lib, and M. Longc, "A semi-fragile watermarking algorithm for authenticating 2D CAD engineering graphics based on log-polar transformation," Comput.-Aided Des., vol.42, Issue 12, pp.1207–1216, Dec. 2010.
[5] R. Ohbuchi, H. Ueda, and S. Endoh, "Robust watermarking of vector digital maps," Proc. IEEE International Conference on Multimedia and Expo (ICME), vol.1, pp.577–580, 2002.
[6] M. Vogit and C. Busch, "Watermarking 2D-vector data for geographical information systems," Proc. SPIE, Security and Watermarking of Multimedia Content, vol.4675, pp.621–628, San Jose, USA, 2002.
[7] C.Y. Shao, H.L. Wang, X.M. Niu, and X.T. Wang, "A shape-preserving method for watermarking 2D vector maps based on statistic detection," IEICE Trans. Inf. & Syst., vol.E89-D, no.3, pp.1290–1293, March 2006.
[8] X.-M. Niu, C.-Y. Shao, and X.-T. Wang, "GIS watermarking: Hiding data in 2D vector maps," Studies in Computational Intelligence, vol.58, pp.123–155, 2007.
[9] J. Aybet, H. Al-Saedy, and M. Farmer, "Watermarking spatial data in geographic information systems," Communications in Computer and Information Science, vol.45, pp.18–26, 2008.
[10] C. Wang, Z. Peng, Y. Peng, L. Yu, J. Wang, and Q. Zhao, "Watermarking geographical data on spatial topological relations," Multimedia Tools and Applications, Online First, May 2010.
[11] B. Wu, W. Wang, Z. Peng, D. Du, and C. Wang, "Design and implementation of spatial data watermarking service system," Geo-Spatial Information Science, vol.13, no.1, pp.40–48, 2010.
[12] A. Li, Y. Chen, B. Lin, W. Zhou, and G. Lv, "Review on copyright wmarking techniques of GIS vector data," Proc. International

Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP '08), pp.989–993, 2008.
[13] C. López, "Watermarking of digital geospatial datasets: A review of technical, legal and copyright issues," Int. J. Geographical Information Science, vol.16, no.6, pp.589–607, Sept. 2002.
[14] V. Solachidis and I. Pitas, "Watermarking polygonal lines using fourier descriptors," IEEE Comput. Graph. Appl., vol.24, no.3, pp.44–51, May 2004.
[15] K. Wang, G. Lavoué, F. Denis, and A. Baskurt, "Three-dimensional meshes watermarking: Review and attack-centric investigation," Information Hiding, LNCS, vol.4567, pp.50–64, 2007.
[16] S.-H. Lee and K.-R. Kwon, "A watermarking for 3D-mesh using the patch CEGIs," Digit. Signal Process., vol.17, Issue 2, pp.396–413, March 2007.
[17] S.-H. Lee and K.-R. Kwon, "Mesh watermarking based projection onto two convex sets," Multimedia Systems, vol.13, no.5-6, pp.323–330, Feb. 2008.
[18] S.-H. Lee and K.-R. Kwon, "3D keyframe animation watermarking based on orientation interpolator," IEICE Trans. Inf. & Syst., vol.E90-D, no.11, pp.1751–1761, Nov. 2007.
[19] E.J. Delp, "Multimedia security: The 22nd century approach," Multimedia Syst., vol.11, no.2. pp.95–97, Oct. 2005.
[20] A. Swaminathan, Y. Mao, and M. Wu, "Robust and secure image hashing," IEEE Trans. Information Forensics and Security, vol.1, no.2, pp.215–230, June 2006.
[21] V. Monga and M.K. Mhcak, "Robust and secure image hashing via non-negative matrix factorizations," IEEE Trans. Information Forensics and Security, vol.2, no.3, Part 1, pp.376–390, Sept. 2007.
[22] B. Coskun, B. Sankur, and N. Memon, "Spatio-temporal transform based video hashing," IEEE Trans. Multimedia, vol.8, no.6, pp.1190–1208, Dec. 2006.
[23] C. De Roover, C. De Vleeschouwer, F. Lefebvre, and B. Macq, "Robust video hashing based on radial projections of key frames," IEEE Trans. Signal Process., vol.53, no.10, Part 2, pp.4020–4037, Oct. 2005.
[24] K. Tarmissia and A.B. Hamza, "Information-theoretic hashing of 3D objects using spectral graph theory," Expert Systems with Applications, vol.36, Issue 5, pp.9409–9414, July 2009.
[25] S.-H. Lee, E.-J. Lee, and K.-R. Kwon, "Robust 3D mesh hashing based on shape features," IEEE International Conference on Multimedia and Expo, pp.1040–1043, July 2010.
[26] S.-H. Lee and K.-R. Kwon, "Robust 3D mesh model hashing based on feature object," Digit. Signal Process., vol.22, Issue 5, pp.744–759, Sept. 2012.
[27] E. Kreyszig, Differential Geometry, Dover Publications, New York, 1991.
[28] Differential geometry of curves, http://en.wikipedia.org, Accessed to June 2011.
[29] M. Hofer, G. Sapiro, and J. Wallner, "Fair polyline networks for constrained smoothing of digital terrain elevation data," IEEE Trans. Geosci. Remote Sens., vol.44, no.10, pp.2983–2990, Oct. 2006.
[30] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Proc. Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp.1027–1035, 2007.
[31] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, 1981.
[32] Y. Chen and M.R. Gupta, "EM demystified: An expectation-maximization tutorial," UWEE Technical Report, UWEETR-2010-0002, Feb. 2010.
[33] J.A. Bilmes, "A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden Markov models," Tech. Rep. TR-97-021, International Computer Science Institute, 1998.
[34] L.J. Heyer, S. Kruglyak, and S. Yooseph, "Exploring expression data: Identification and analysis of coexpressed genes," Genome Research, vol.9, pp.1106–1115, 1999.
[35] AutoCAD, http://www.autodesk.com, Accessed to June 2011.
[36] National Geographic Information Institute, http://www.ngi.go.kr, Accessed to June 2011.

Suk-Hwan Lee received a B.S., a M.S., and a Ph. D. degree in Electrical Engineering from Kyungpook National University, Korea in 1999, 2001, and 2004 respectively. He worked at Electronics and Telecommunications Research Institute in 2005. He is currently an associate professor in Department of Information Security at Tongmyong University, which he started in 2005. He works as an editor of Korea multimedia society journal and is a member of IEEE, IEEK and also is an officer of IEEE R10 Changwon section. His research interests include multimedia signal processing, multimedia security, digital signal processing, bio security, and computer graphics.

Seong-Geun Kwon received the B.S., and M.S., and Ph.D degrees in Electronics Engineering in Kyungpook National University, Korea in 1996, 1998, and 2002, respectively. He worked at Mobile Communication Division of Samsung Electronics from 2002 to 2011 as a Senior Engineer and Project Manager. He is currently a assistance professor in department of Electronics Engineering at Kyungil University. His research interests include multimedia security, mobile VOD, mobile broadcasting (T-DMB, S-DMB, FLO, DVB-H), and watermarking.

Ki-Ryong Kwon received the B.S., M.S., and Ph.D. degrees in Electronics Engineering from Kyungpook National University in 1986, 1990, and 1994 respectively. He worked at Hyundai Motor Company from 1986–1988 and at Pusan University of Foreign Language form 1996–2006. He is currently a professor in Department of IT Convergence and Application Engineering at the Pukyong National University. He visited University of Minnesota at 2000–2001 and Colorado State University at 2011–2012 with visiting professor in USA. He is currently the General Affair and Organizing Committee/Vice President in Journal of Korea Multimedia Society. Also is a director of IEEE R10 Changwon section. His current research interests are in the area of digital image processing, multimedia security and digital watermarking, wavelet transform.