Robust Hashing of Vector Data Using Generalized Curvatures of Polyline*

Suk-Hwan LEE\textsuperscript{[a]}, Member, Seong-Geun KWON\textsuperscript{[b]}, and Ki-Ryong KWON\textsuperscript{[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)\cite{1} and CAD (computer-aided design)\cite{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\cite{3, 4}, GIS digital maps\cite{5–13}, vector graphics\cite{14}, and 3D model\cite{15–18}. On the other hand, E.J. Delp\cite{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\cite{20, 21}, videos\cite{22, 23}, and 3D models\cite{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\cite{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\cite{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...
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 the 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, $\chi_1$, $\chi_2$, 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, $I$ be a non-empty interval of real numbers, and $t$ be any real number in $I$. Then, a parametric curve $\gamma$ is defined as a vector-valued function, $\gamma: I \rightarrow \mathbb{R}^2$, that is $r$ times continuously differentiable in $\mathbb{R}^2$, which is regular of order $m$; $|\gamma'(t), \gamma'(t), \cdots, \gamma^{(m)}(t)|$, $m \leq r$, $t \in I$. The Frenet frame of a parametric curve $\gamma$ is the set of $m$ orthonormal vectors, $\{e_1(t), \cdots, e_m(t)\}$, 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 $\gamma(t)$, that are used for describing a curve locally. Each orthonormal vector $e_j(t)$ can be reconstructed from derivatives of $\gamma(t)$ using the Gram-Schmidt orthogonalization algorithm: $e_j(t) = \bar{e}_j(t)/\|\bar{e}_j(t)\|$ where $\bar{e}_j(t) = \gamma^{(j)}(t) - \sum_{i=1}^{j-1} \langle \gamma^{(i)}(t), e_i(t) \rangle e_i(t)$. The second curvature is defined as the real function, $\chi_2(t) = \langle e_1(t), e_{1+1}(t) \rangle/\|\gamma'(t)\|^2$. 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 $p$ with $n+1$ vertices, $\{v_0, \cdots, v_n\}$, which is piecewise continuous, the Frenet vectors $e_j[k]$ for vertices $v_k$ are

$$e_j[k] = \frac{\bar{e}_j[k]}{\|\bar{e}_j[k]\|}, \quad k \in [0, n]$$

where $e_1[k] = \gamma'(k)/\|\gamma'(k)\|$ and $\bar{e}_j[k] = \gamma^{(j)}[k] - \sum_{i=1}^{j-1} \langle \gamma^{(i)}[k], e_i[k] \rangle e_i[k]$. The first Frenet vector $e_1[k]$ is the unit tangent vector that is defined at each point $v_k$ of a parametric curve $\gamma$; the second Frenet vector $e_2[k]$ is the unit curvature vector or the unit normal vector that describes the deviation of the curve from a straight line. The two vectors $e_1[k]$ and $e_2[k]$ at a point $v_k$ define the osculating plane at this point. The third Frenet vector $e_3[k]$ is the binormal vector, which is always orthogonal to $e_1[k]$ and $e_2[k]$. We obtain the first curvature $\chi_1[k]$ using $e_1[k]$ and $e_2[k]$, and the second curvature $\chi_2[k]$ using $e_2[k]$ and $e_3[k]$.

$$\chi_1[k] = \frac{\langle e_1[k], e_2[k] \rangle}{\|\gamma'(k)\|}, \quad \chi_2[k] = \frac{\langle e_2[k], e_3[k] \rangle}{\|\gamma'(k)\|}$$

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


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; \( \mathbf{M} = \{ \mathbf{M}_i | i \in [1, N] \} \). 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; \( \mathbf{p}_{ij} = \{ \mathbf{v}_{ij,k} | k \in [1, N_{ij}] \} \), which is a \( j \)th polyline in the \( i \)th 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 \( \gamma_i \) is the ratio of the polyline number to the total number of other primitives and \( \rho_i \) is the polyline density on each layer \( \mathbf{L}_i \).

Let \( \sum_{k=1}^N N(\mathbf{L}_k) \) be the number of polylines in a layer \( \mathbf{L}_k \) and \( \eta_i \) be the ratio of the accumulated polyline number, \( \sum_{i=1}^k N(\mathbf{L}_i) \), in \( i \) layers (\( i \leq N \)) 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 \( \eta_i \) is above \( Th_\eta \). Therefore, the selected layer \( \mathbf{L}_* \) is defined by

\[
\mathbf{L}_* = \{ \mathbf{L}_1, \cdots, \mathbf{L}_{i*}, \cdots, \mathbf{L}_{N_L} \},
\]

\( \forall \eta_i > Th_\eta \) and \( \gamma_i \rho_i > \gamma_{i+1} \rho_{i+1}, \ i \in [1, N_L] \)

This means that the number of polylines in \( \mathbf{L}_* \) is more than \( Th_\eta \times 100[\%] \) of the number of total polylines. If \( Th_\eta \) is small, the number of the selected layers and hash length increase; however, the robustness decreases because insignificant layers are selected. In contrast, if \( Th_\eta \) 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 \( Th_\eta \) 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 \( \eta \) 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 \( \mathbf{p}_{ij} = \{ \mathbf{v}_{ij,1}, \cdots, \mathbf{v}_{ij,N_{ij}} \}, N_{ij} < 4 \). The refinement process links \( \mathbf{p}_{ij} \) 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 \( \mathbf{p}_{ij} \). If \( \mathbf{p}_{ij} \) has no polylines within the range of the average polyline length, \( \mathbf{p}_{ij} \) will not be selected to the curvature. Figure 3 illustrates an example that a current line \( \mathbf{p}_{ij} \) is refined by the sequential
Fig. 3 An example of polyline refinement of \( p_{ij} \) to link between the nearest polylines and \( p_{ij}' \).

The linkage of left and right nearest polylines, \( p_{ij} \) and \( p_{ij}' \).

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 \( L_i \). For a smooth curve \( \Upsilon(t) \), defined in some parameter interval \([a, b]\), the linearized bending energy (cubic spline energy) is defined by \( E[\Upsilon(t)] = \int_0^b \|\Upsilon^{(2)}(t)\|^2 dt \). Here, \( \Upsilon^{(2)} \) denotes the second derivative vector of the curve \( \Upsilon(t) \) with respect to the curve parameter \( t \). A polyline \( p_{ij} \) as a discrete curve possesses a discrete linearized bending energy \( E[p_{ij}] \), which is called “bending energy” for short.

\[
E[p_{ij}] = \sum_{k=2}^{N_{ij} - 1} ||\Delta^2 v_{ijk}||^2, 
\]

\[
\Delta^2 v_{ijk} = v_{ij,k-1} - 2v_{ijk} + v_{ij,k+1} 
\]

We then cluster all polylines into \( N_{H} \) groups, which is the length of hash, using the normalized bending energy \( \bar{E}[p_{ij}] \)

\[
\bar{E}[p_{ij}] = \frac{E[p_{ij}]}{\sum_{j=1}^{N_{H}} E[p_{ij}]} 
\]

We denote the set of \( N_{H} \) groups as \( G_i = \{G_{i,n} | n \in [1, N_{H}] \} \). 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.

Let the distribution of \( E[p_{ij}] \) be the sum of Gaussian distributions weighted by \( \omega_n; \{\phi(\mu_n, \Sigma_n)\}_{n=1}^{N_{H}} \), where \( \mu_n \) is the mean and \( \Sigma_n \) is the covariance. A weight \( \omega_n \) has the property of \( \omega_n > 0 \) and \( \sum_{n=1}^{N_{H}} \omega_n = 1 \). Let \( \theta_n \) be parameters of weight, mean, and covariance of \( n \)th group; \( \theta_n = (\omega_n, \mu_n, \Sigma_n) \). We randomly select \( N_{H} \) initial parameters \( \{\theta_{n_0}^{(0)}\}_{n_0 \in [1, N_{H}]} \) in the selected layer \( L_i \) 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.

\[
G_{i,n} = \{p_{ij} : Pr(\bar{E}[p_{ij}] | \theta_n) > Pr(\bar{E}[p_{ij}] | \theta_{n'}), \forall n' \neq n \in [1, N_{H}] \}
\]

This set of parameters \( \Theta = \{\theta_n\}_{n \in [1, N_{H}]} \) is stored to extract the hash. All polylines in a layer \( L_i \) are clustered to \( N_{H} \) groups; \( \{G_{i,n}\}_{n \in [1, N_{H}]} \).

3.3 Hash Feature Value

3.3.1 Polylines Curve

The average of the first and second curvatures, \( \chi_1(p_{ij}) \) and \( \chi_2(p_{ij}) \), of polylines in the selected layer \( L_i \) are calculated from the following equation derived from Eq. (2).

\[
\chi_1(p_{ij}) = \sum_{k=3}^{N_{ij}} \frac{\chi_1[k]}{N_{ij} - 3}, \chi_2(p_{ij}) = \sum_{k=3}^{N_{ij}} \frac{\chi_2[k]}{N_{ij} - 3} 
\]

As shown in Fig. 4, we use the three vertices, \( v_{ij,1}, v_{ij,2}, v_{ij,3} \), for the initial points because the third Frenet vector \( e_3[k] \) is to be twice differentiable. The second and third Frenet vectors, \( e_2[k] \) and \( e_1[k] \) at a vertex \( v_{ijk}(k \in [3, N_{ij}]) \) can be obtained as follows:

\[
e_2[k] = \frac{\bar{e}_2[k]}{||\bar{e}_2[k]||}, e_3[k] = \frac{\bar{e}_3[k]}{||\bar{e}_3[k]||} 
\]

where \( \bar{e}_2[k] = \gamma_0^{(2)}[k] - \gamma_1^{(1)}[k], e_1[k] > e_2[k], \bar{e}_3[k] = \gamma_0^{(3)}[k] - \sum_{j=1}^{2} \gamma_{j}^{(1)}[k], e_2[k] > e_3[k] \).

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

\[
\begin{bmatrix}
\bar{e}_1'[k] \\
\bar{e}_2'[k]
\end{bmatrix} = \begin{bmatrix}
0 & \chi_1[k] \\
-\chi_1[k] & 0
\end{bmatrix} \begin{bmatrix}
\bar{e}_1[k] \\
\bar{e}_2[k]
\end{bmatrix} 
\]

As described above, we calculate pairs of the first and second average curvatures, \( (\bar{\chi}_1(p_{ij}), \bar{\chi}_2(p_{ij})) \), of all polylines in each group.
We generate two Gaussian random matrices, \( R_{Ni} \) where \( Ni \) is the group value for the first group and the second group value. Thus, the first ages of the first and second curvatures, which are called the group feature matrix \( F \) of size \( NH \times NL \), are defined as follows.

\[
F = \begin{bmatrix} x_{1,n}^{(1)} & x_{2,n}^{(1)} & \ldots & x_{NL,n}^{(1)} \\ x_{1,n}^{(2)} & x_{2,n}^{(2)} & \ldots & x_{NL,n}^{(2)} \end{bmatrix}
\]

for \( n_{Li} \in G_{i,n} \).

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

\[
X^{(1)} = \{x_{1,n}^{(1)}|i \in [1,N_L], n \in [1,N_H]\},
\]

\[
X^{(2)} = \{x_{2,n}^{(2)}|i \in [1,N_L], n \in [1,N_H]\}.
\]

We generate two Gaussian random matrices, \( R_1, R_2 \) with mean \( n_R \) and variance \( \sigma_R^2 \) of size \( NL \times NH \), and then calculate the group feature matrix \( F \) of size \( NL \times NH \), which is defined by the products of \( [X^{(1)}]^T \) and \( R_1 \), and of \( [X^{(2)}]^T \) and \( R_2 \).

\[
F = [X^{(1)}]^T R_1 + \alpha [X^{(2)}]^T R_2
\]

The feature value is written as

\[
f_{m_1,n_2} = \sum_{i=1}^{N_L} x_{i,m_1}^{(1)} r_{1,i,n_2} + \alpha \sum_{i=1}^{N_L} x_{i,m_1}^{(2)} r_{2,i,n_2}
\]

\( \forall n_1, n_2 \in [1,N_H] \)

\([X]^T\) is the transposed matrix of \( X \). The intensity \( \alpha \) equalizes the dynamic range of \( x_{1,n}^{(1)} \) and \( x_{2,n}^{(2)} \).

\[
\alpha = \max\{|x_{1,n}^{(1)}|/\max|x_{2,n}^{(2)}|\}
\]

3.4 Hash Generation

The binary hash \( H \) is generated from the binarization of the group feature value \( F \).

\[
h_{m_1,n_2} = \begin{cases} 0, & f_{m_1,n_2} < Th_h \\ 1, & f_{m_1,n_2} \geq Th_h \end{cases}
\]

The threshold \( Th_h \) is determined for the binary hash as a Bernoulli distribution with \( Pr[h_{m_1,n_2} = 1] \approx Pr[h_{m_1,n_2} = 0] \approx 0.5 \), as follows.

1) Select an initial threshold \( Th_h^{(0)} \) randomly in the dynamic range of \( f_{m_1,n_2} \).

2) Classify feature values to two groups \( G_1, G_2 \) using \( Th_h^{(t)} \) \((t \geq 0)\), \( G_1 = \{f_{m_1,n_2}|f_{m_1,n_2} \geq Th_h^{(t)}\}, G_2 = \{f_{m_1,n_2}|f_{m_1,n_2} < Th_h^{(t)}\} \).

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 \( H' \) in the received vector data model \( M' \) from a process similar to that of hash generation, and determines the authentication of \( M' \) using the normalized Hamming distance \( d(H,H') \).

\[
d(H,H') = \frac{1}{NH} \sum_{n_1} \sum_{n_2} |h_{m_1,n_2} - h_{m_1,n_2}'|
\]

\[
M' \rightarrow \begin{cases} \text{Auth.} & d(H,H') < \epsilon \\ \text{NotAuth} & d(H,H') \geq \epsilon \end{cases}
\]

\( d(H,H') \) 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 \( \epsilon \). The test extracted 5,000 hashes in 50 models using 100 keys, and calculated \( d(H,H') \) between the original and attacked hashes and measured the true positive (TP) and false positive (FP) rates while varying \( \epsilon \) from 0 to 0.5 in steps of 0.05. TP is the probability \( pr_{TP} = Pr[d(H,H') < \epsilon] \) that the hash \( H' \) of attacked model \( M' \) is authenticated to the original model \( M \), and FP is the
Table 1  The number of layers and polylines, differential entropy, and processing time of typical vector data models.

<table>
<thead>
<tr>
<th>Test model</th>
<th>Total number</th>
<th>Selected number</th>
<th>Polyline ratio [%] (N/T×100)</th>
<th>Differential entropy</th>
<th>Processing Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drawing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block and Tables</td>
<td>16</td>
<td>76</td>
<td>2</td>
<td>55</td>
<td>72.4</td>
</tr>
<tr>
<td>Db</td>
<td>29</td>
<td>1,737</td>
<td>2</td>
<td>1,351</td>
<td>77.8</td>
</tr>
<tr>
<td>Hummer Elev.</td>
<td>30</td>
<td>1,492</td>
<td>1</td>
<td>1,492</td>
<td>100.0</td>
</tr>
<tr>
<td>Stadium Elev.</td>
<td>52</td>
<td>5,442</td>
<td>2</td>
<td>3,916</td>
<td>71.9</td>
</tr>
<tr>
<td>Tablet</td>
<td>5</td>
<td>1,458</td>
<td>2</td>
<td>1,201</td>
<td>82.4</td>
</tr>
<tr>
<td>Taisei Detail Plan</td>
<td>37</td>
<td>2,288</td>
<td>1</td>
<td>2,288</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>633</td>
<td>2</td>
<td>481</td>
<td>75.9</td>
</tr>
<tr>
<td>1:1,000</td>
<td>74</td>
<td>2,489</td>
<td>3</td>
<td>1,851</td>
<td>74.4</td>
</tr>
<tr>
<td>1:25,000</td>
<td>116</td>
<td>18,911</td>
<td>6</td>
<td>16,361</td>
<td>86.5</td>
</tr>
<tr>
<td>1:250,000</td>
<td>29</td>
<td>2,001</td>
<td>4</td>
<td>1,530</td>
<td>76.5</td>
</tr>
</tbody>
</table>

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.

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

$p_{FD}$ is the probability that $d(H, H')$ of an original hash $H$ and an attacked hash $H'$ at any attack intensity $\delta$ is above the threshold $\epsilon$. We calculated $p_{FD}$ by iteratively performing random key generation and hash extraction until $d(H, H') > \epsilon, (\epsilon = 0.15)$ at the attack intensity $\delta$. 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 \( E' = 1/s \). 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 \( 6 \times 10^{-4} \) to \( 4.2 \times 10^{-2} \) up to 50% cropping of the data and \( 1.5 \times 10^{-4} \) to \( 2.9 \times 10^{-3} \) 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.

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 \( 6.0 \times 10^{-4} \) to \( 4.3 \times 10^{-2} \) with 10–30% of the polylines trimmed and \( 7.5 \times 10^{-4} \) to \( 2.9 \times 10^{-2} \) with 10-30% of the polylines broken. But it is somewhat high from \( 1.1 \times 10^{-1} \) to \( 2.1 \times 10^{-1} \) 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.

![Fig. 8](image1.png) **Fig. 8** The front of first floor of (a) Hummer Elevation drawing, (b) 30% data cropped drawing, (c) 30% data copied drawing.

![Fig. 9](image2.png) **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.
is not a

4.1 Tensions in vector data editing tools. In addition, our algorithm extending, and simplifying, and robust up to a specific range of the normalized bending energy and the group values. Ac-

pFDaption, as shown in Fig. 7 (f), reveals that the polylines using the subsample. The result of the simplifica-

tion, as shown in Fig. 7 (e), reveals that the $p_{FD}$ is very low from $1.4 \times 10^{-4}$ to $8.5 \times 10^{-3}$ 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 de-

pend not on the length but on the curve. Accordingly, we verified that our hash is robust to extending up to 50%.

4.1.4 Extending

The extending process extends any polyline for being con-

nected to the corner of a user-defined polyline. In an ex-

tending 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), re-

eals that the $p_{FD}$ is very low from $1.4 \times 10^{-4}$ to $8.5 \times 10^{-3}$ 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 de-

pend 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 $1.6 \times 10^{-3}$ to $2.5 \times 10^{-2}$ with $10-50\%$ of the polylines sim-

plified. The simplification causes the curve to lack smooth-

ness, while preserving the shape of the polylines. It some-

what affects the first and second curvatures but affects less the normalized bending energy and the group values. Ac-

cordingly, 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, ex-

tending, 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.

<table>
<thead>
<tr>
<th>Category</th>
<th>Model Uniq.</th>
<th>Key Uniq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Pr[d(H_1, H_2) &gt; 0.5 - \varepsilon]$</td>
<td>0.956987</td>
<td>0.98</td>
</tr>
<tr>
<td>$Pr[\varepsilon &lt; d(H_1, H_2) &lt; 0.5 - \varepsilon]$</td>
<td>0.043012</td>
<td>0.02</td>
</tr>
<tr>
<td>$Pr[d(H_1, H_2) &lt; \varepsilon]$</td>
<td>0.000001</td>
<td>0.00</td>
</tr>
</tbody>
</table>

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 $H_1, H_2$ generated in different models $M_1, M_2$ with the same key $K$ are statistically very different.

$$Pr[H_1 \neq H_2] \approx 1$$ (19)

where $H_1 = hash(M_1, K), H_2 = hash(M_2, K)$

Key uniqueness refers to the fact that two hashes $H_1, H_2$ generated in different keys $K_1, K_2$ with the same model $M$ are statistically very different.

$$Pr[H_1 \neq H_2] \approx 1$$ (20)

where $H_1 = hash(M, K_1), H_2 = hash(M, K_2)$

Thus, the normalized Hamming distance $d(H_1, H_2)$ of two hashes, in model and key uniqueness experiments, must ap-

proach 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. $Pr[d(H_1, H_2) \geq 0.5 - \varepsilon]$ indicates that the two hashes are statistically very different, and $Pr[d(H_1, H_2) < \varepsilon]$ indicates that the two hashes are statistically very similar. In the results for model unique-

ness, $Pr[d(H_1, H_2) \geq 0.5 - \varepsilon]$ is 0.95 and $Pr[d(H_1, H_2) < \varepsilon]$ 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, $Pr[d(H_1, H_2) \geq 0.5 - \varepsilon]$ is 0.98 and $Pr[d(H_1, H_2) < \varepsilon]$ 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 en-

tropy $H(f) = - \int p(f) \log(p(f)) df$ from a Gaussian distribution. The sum of two variables, $A = \sum_{i=1}^{N_A} x_{i,A}^{(1)} r_{1,i,n_1}$ and $B = \alpha \sum_{i=1}^{N_A} x_{i,A}^{(2)} r_{2,i,n_2}$, can be rewritten as the sum of two variables, $A = \sum_{i=1}^{N_A} x_{i,A}^{(1)} r_{1,i,n_1}$ and $B = \alpha \sum_{i=1}^{N_A} x_{i,A}^{(2)} r_{2,i,n_2}$, in Eq. (13). Given that two random values $r_1, r_2$ for the first and second group, $x_{i=1}^{(1)}$ and $x_{i=1}^{(2)}$, are $(m_1, \sigma_1^2)$ and $(m_2, \sigma_2^2)$, the two variables $A, B$ are normally distributed, $(m_A, \sigma_A^2) = (\sum_{i=1}^{N_A} x_{i,A}^{(1)} r_{1,i,n_1}, \sum_{i=1}^{N_A} x_{i,A}^{(1)} r_{1,i,n_1}^2).$ Therefore, $p(f)$ is normally distributed with a mean $m_f$ and a variance $\sigma_f^2$. 

![Image](https://example.com/image1.png)

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

up to 30% trimming and breaking.
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


Suk-Hwan Lee received a B.S., a M.S., and a Ph. D. degree in Electrical Engineering from 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 an assistant professor in department of Electronics Engineering at Kyungil University. His research interests include multimedia security, multimedia signal processing, bio security, and computer graphics.

Seong-Geun Kwon received the B.S., 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 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 research interests include multimedia signal processing, multimedia security, mobile VOD, mobile broadcasting (T-DMB, SFBC, 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.