Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal, 2026, 20(7), 139–156
https://doi.org/10.12913/22998624/218894
ISSN 2299-8624, License CC-BY 4.0

Received: 2026.03.27
Accepted: 2026.05.15
Published: 2026.06.01

Geometry-aware fragile watermarking with cryptographic functions for authenticity verification of glTF 3D models

Marcin Matczuk[1*], Mateusz Traczyński[2]

[1] Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland
[2] Department of Quantitative Methods in Management, Lublin University of Technology, Nadbystrzycka 38, 20-618 Lublin, Poland
* Corresponding author's e-mail: m.matczuk@pollub.pl

ABSTRACT

Modern graphics processing units possess the requisite computing power to efficiently render complex three-dimensional (3D) scenes. The graphics library Transmission Format (glTF) is a file format and a standard that facilitates the efficient transfer of entire virtual worlds between various applications. Despite the growing adoption of the glTF in the context of rendering and transferring 3D content, the area of watermarking within this format remains poorly researched. The article presents a novel scheme for verifying the authenticity of 3D assets in glTF standard. The method is based on a fragile watermarking mechanism and allows for the localisation of sabotage spots. The work employs a hash-based message authentication code (HMAC) and a BLAKE2 cryptographic hash function to generate components of the watermark. An authorial algorithm called the high-density mesh locator (HDML) has been developed. The HDML identifies areas with dense, complex geometry and leverages a *kd*-tree data structure to expedite and optimise spatial queries. The selection of vertices located in regions of high geometric complexity effectively obscures the modifications made. The mechanism for detecting sabotage involves segmenting the model using *k*-means++ algorithm and applying a separate part of the watermark to each segment. The methodology consists of four sequential stages: (1) watermarking various 3D models; (2) making modifications or omitting them; (3) verifying the authenticity of data based on a watermark; and (4) detecting places of interference in the model. The proposed method introduces a minimal level of interference into the geometric structure of the object. This assertion is supported by the obtained Peak Signal-to-Noise Ratio metric of ~210 dB and Hausdorff distance of $\sim 10^{-8}$. The amalgamation of short hashes and the authorial HDML algorithm yields a highly transparent watermark that allows the detection of a single bit change.

Keywords: 3D model, fragile watermarking, 3D mesh watermarking, hash-based message authentication code, *k*-means++, *kd*-tree, cybersecurity, integrity protection.

INTRODUCTION

In the contemporary era, remote work has become a prevalent phenomenon, particularly within the IT industry, where professionals from diverse geographical locations collaborate on projects. In such cases, a significant volume of data is transferred between team members and the primary server. Consequently, it is imperative to utilise appropriate tools for verification of authenticity, as digital data, lacking adequate security, is susceptible to easy falsification or malicious content insertion [1]. The growing number of online scams and the increase in digital crime clearly indicate how serious the problem is [2,3].

Thanks to the rapid development of computer graphics, three-dimensional (3D) models have expanded their range of applications substantially [4]. Cultural heritage objects [5,6], experimental measurements [7], clinical datasets [8], and forensic scenes [9] are recorded in the form of 3D models. Moreover, the number of datasets containing 3D motion data [10] is growing thanks to advances in motion capture technology [11,12].

Such sensitive data must be properly secured. It is therefore clear that the area of cybersecurity also applies to 3D resources.

Presently, among the many standards intended for the storage of 3D assets, the graphics library Transmission Format (glTF) is one of the most universal, all-inclusive, and commonly adopted. It allows not only individual models to be stored, but also entire virtual scenes, along with information such as camera settings, transformation matrices, light properties, and animation data.

The security of data is ensured not only by cryptographic methods, but also by steganographic methods, which are employed extensively [13,14]. The primary distinction between steganography and cryptography is the former's emphasis on concealing the act of communication and data exchange, in contrast to the latter's focus on encrypting the content of the message. The focus of steganographic techniques is primarily on the imperceptible embedding of data in audio [15], video [16], images [17] and 3D models [18]. Given that the fundamental principle of modern steganography is to embed information in digital resources in the least noticeable way possible, it is an ideal solution for watermarking 3D assets.

Watermarking of 3D resources is an advanced technique that involves attaching a small amount of additional information to an object. This additional information is only visible after applying the appropriate extraction procedure. Watermarks can be categorised into three primary classifications [19]: robust – resistant to various types of modifications, used mainly to confirm copyright, fragile – susceptible to all forms of interference, used to verify data authenticity and detect unauthorised changes, and semi-fragile – combining the characteristics of the previous two. The watermarking process itself can be reversible or irreversible. The goal of reversible steganography is to introduce modifications to an object in a way that enables the complete removal of these changes. In other words, the object can be fully restored to its pre-watermarking state [20,21].

The motivation for writing this article is to present a pioneering method of verifying the authenticity of 3D assets based on the fragile watermarking mechanism. The objective of this study is to develop a verification method for files saved in the glTF standard. Despite its growing popularity and versatility, glTF has not attracted much interest in the scientific community. Due to its intricate structure, the glTF format possesses considerable steganographic potential, which has thus far remained largely unexplored. To date, only one study has been conducted strictly on watermarking models stored in the glTF standard [22].

The method delineated in this article is characterised by a high degree of transparency of the changes introduced, the ability to detect modifications of even a single bit, and the ability to locate points of attack on 3D resources. The development of such a versatile mechanism was achieved through the implementation of an amalgamation of techniques, including the use of Hash-based Message Authentication Code (HMAC), BLAKE2 cryptographic hash function, and an authorial algorithm called the High-Density Mesh Locator (HDML), as well as model segmentation using the *k-means++* algorithm. The HDML is designed to identify areas characterised by dense and complex geometric structures.

RELATED WORKS

Nowadays, machine learning models and widely known artificial intelligence (AI) are prevalent in scientific research. These techniques have been spreading across almost every scientific domain, including steganography [23,24]. Currently, many watermarking methods utilizing AI have been developed [25,26]. Moreover, a novel research area is currently emerging in the field of watermarking, focusing on the watermarking of neural networks [27,28].

Another intriguing avenue for exploration pertains to the implementation of watermarking techniques in oblique images (i.e., oblique 3D models), a method that has found application, inter alia, in the domain of 3D cartography [29]. In [30] a novel approach for marking 3D models of oblique photography (3DMOP) was proposed, which is founded on the division of the point cloud that constitutes the 3DMOP into clusters. Subsequently, the centroid and feature points are calculated for each cluster. The watermark is embedded in each cluster, with the location of each cluster determined on the basis of the distance between the centroid and the feature points. Jiao Y. et al. also employed texture coordinates to embed the watermark.

Among the popular watermarking methods are approaches based on various types of model geometry clustering [31, 33]. In [32] a 3D mesh

watermarking algorithm employing the fuzzy logic [34] ang fuzzy c-means clustering (FCM) [35] was proposed. The FCM was utilised to assess the suitability of the vertices of the model for watermark embedding, with this assessment being based on feature vectors. The feature vector is constructed as a set of angles between the normal vectors and the average normal vector of triangular faces forming a ring around a given vertex. In this study, two approaches presented to watermark embedding. The first approach is based on local statistical measures, such as the mean and standard deviation, which are used to modify vertex values in a way that represents a secret watermark. The second approach involves using jumbled insertion planning. Article [32] is an extension of [36], which presented a non-blind robust watermarking approach for 3D mesh models based on *k*-means clustering. The method that utilises FCM yields results that are more optimal than the *k*-means approach.

A significant number of researchers are investigating the application of various transforms to convert the spatial representation of a model into another form [37–39]. Kozieł and Malomuzh [37] incorporated the Hash-based Message Authentication Code as a watermark, meticulously concealing it within the coefficients of the discrete wavelet transform (DWT). Selected components of selected vertices were subjected to DWT transformation. Following this, the HMAC code bits were used to modulate the DWT coefficients. In the next step of the process, the inverse DWT transformation was performed, and the modified vertex components were embedded back into the model.

In [38] the discrete wavelet transform for three-dimensional meshes (DT3D) was employed, facilitating the hierarchical decomposition of the model into successive levels of representation with progressively simplified geometry. Following the DT3D transform, the watermark integrated, which is represented by a sequence of alphanumeric characters, into the coefficients associated with low frequencies. The embedding process entails a modification of the values of these coefficients, a modification that is dependent on the values of individual bits of the watermark.

The primary focus of [39] is the utilisation of Clifford Multiwavelet Transform to embed copyright data in the multiresolution domain, a technique that facilitates a substantial augmentation in the size of the embedded watermark. The watermark in this approach consists of the following components: mesh description, source mesh signature (produced using SHA512), and a logo encrypted using RSA.

The approach proposed in [40] based on the analysis of principal curvatures, Gaussian curvature, mean curvature, and edge distribution of the model. These steps are taken to estimate the topological and geometric characteristics of the vertices. In the proposed method, vertices characterised selected by negative mean curvature and an odd number of edges in their nearest neighbourhood. Bits of the text watermark are embedded in such locations.

Articles [41,42] introduce interesting watermarking methods based on different types of interpolation. In [41], the authors propose first transforming the watermark carrier using Lagrange interpolation. Subsequently, the interpolated object (image or model) is divided into five-point segments on which Bézier curves are constructed. The embedding of a bit of information is performed on the interpolated values by selecting the nearest point on the Bézier curve, rounded to an integer, such that its least significant bit coincides with the bit to be embedded. The interpolated values are then replaced with the corresponding Bézier curve values.

Work [42] investigates interpolation-based steganography for raster images. The authors primarily evaluate two methods: improved neighbor mean interpolation and standard neighbor mean interpolation. In the study, the regular–singular method was used to estimate the percentage of additional information detectable in the containers.

Solutions using cryptographic hashes are also emerging. In [43] the watermark is generated using a hashing function based on the Arnold transform and singular value decomposition. The embedding process conceals the watermark by modifying vertex positions, depending on the valence or degree of the surrounding faces.

This article introduces an authorial HDML algorithm that identifies areas of high curvature. It is therefore worth relating work [44], in which the mesh structural distortion measure (MSDM) was introduced as an equivalent of the structural similarity index measure (SSIM) for three-dimensional objects. The calculation of the curvature of the model is a prerequisite for determining the MSDM. Lavoué et al. estimated the curvature for each vertex of the 3D model in the form of a curvature tensor, implementing the

approach described in [45], based on the concept of normal cycles. The calculation of the curvature of the vertex also necessitated the determination of a consistent neighbourhood within a sphere of a fixed radius.

MATERIALS AND METHODS

glTF standard and GLB file extension

The glTF format was developed in response to the increasing complexity and size of virtual 3D scenes. Version 2.0, released in 2017, has become the dominant format in real-time applications due to its lightweight structure, efficiency, and support for modern technologies. The glTF 2.0 standard (hereinafter referred to as glTF) allows for the storage of complete 3D scenes, including complex geometry, materials, light sources and animations, making it a versatile solution for modern rendering engines. The relationship between the components of the glTF is illustrated in Figure 1, where arrows indicate the direction of aggregation relationships.

To improve real-time loading speed, the glTF standard introduced the graphics library binary (.glb) file format, which packages all content into a single binary buffer. Figure 2 shows the structure of a .glb file. The JavaScript object notation (JSON) segment defines the structure and metadata of the binary buffer, which allows for the location and access of individual data.

3D models

The proposed method has undergone rigorous testing on a range of models and 3D assets stored in files with the .glb extension; some of them are presented in Table 1.The models differ in terms of their creation process, level of complexity, and geometric characteristics. For instance, the *gate* model was created using a three-dimensional scanning technique on historical monuments [46], while the *shoe* model was created using photogrammetry [47]. The objects *cannon*, *torch*, and *shoe* were obtained free of charge from https://www.turbosquid.com/, in accordance with the standard license.

3D resources using the glTF standard are mainly created for rendering high-quality virtual scenes. Consequently, a single vertex typically encompasses a greater amount of information than a mere position vector: $\vec{v} \in \mathbb{R}^3$. Additional data include, among other things: normal vectors: $\vec{n} \in \mathbb{R}^3, \|\vec{n}\| = 1$, which are essential for precise lighting calculations; texture coordinates: $(u, v) \in [0.1] \times [0.1]$, required for the correct mapping of materials on the model surface; and tangent vectors: $\vec{t} \in \mathbb{R}^3, \|\vec{t}\| = 1$, which can be used to determine parameters such as torque, angular velocity, or light refraction. These elements are called vertex attributes. Models lacking all attributes were deliberately selected to evaluate the effectiveness of the method across a broad spectrum of cases.

HMAC and BLAKE

Hash-based message authentication code (HMAC) is a mechanism for calculating Message Authentication Code using a cryptographic hash function [48,49] in combination with a secret key. HMAC allows for the verification of data integrity and authenticity and is resistant to forgery, due to the fact that a key is required to generate HMAC, which significantly increases the level of security [50]. Equation 1 shows the general scheme for generating an HMAC code.

$$\begin{aligned} &\mathrm{HMAC}(K, M) = \\ &= H\begin{pmatrix} (K' \oplus \mathrm{opad}) \\ \| H(K' \oplus \mathrm{ipad} \| M) \end{pmatrix} \end{aligned} \tag{1}$$

where: H – cryptographic hash function with a block size of L bytes, K – secret key, K' – secret key after initial processing. If the size of K is greater than L, then K' = H(K), otherwise zeros must be appended to the key to obtain size L, M – encrypted message, $\oplus$ – XOR bitwise operation, higher priority than $\|$ – concatenation, ipad, opad – byte 0×36 and byte 0×5c duplicated L times, respectively.

The computational complexity of HMAC is O(n), as it effectively acts as a wrapper around the underlying hash function. The complexity of the hash function itself depends on the number of data blocks it processes. In Big O notation, this relationship is linear.

The paper also uses the BLAKE2 hash function [51], which is an improved version of the BLAKE algorithm. BLAKE2 is significantly faster than SHA-3, SHA-2, and even MD5, while maintaining a high level of security comparable

to SHA-3 [52]. In the rest of the article, we refer to the code obtained from both BLAKE2 and HMAC as a hash. Technically, HMAC is not a cryptographic hash function, and the word hash refers only to the result returned by HMAC.

Overview of the method

The proposed solution is based on hashes generated using HMAC and BLAKE2, which act as parts of fragile watermark. Due to the fact that the BLAKE2 and HMAC mechanisms used return completely different results even with minimal changes to the input data (e.g., modification of a single bit), they are suitable for detecting the smallest modifications made to the model.

The developed method consists of the following steps: (1) determining the optimal locations for hiding the watermark using the authorial HDML algorithm; (2) zeroing the least significant bit (LSB) in the coordinates of attributes associated with the vertices intended for embedding watermarks; (3) dividing the model mesh into segments using k-means++ clustering; (4) generating watermark components based on vertex attributes and the obtained segments; (5) embedding the watermark in the model structure; (6) verifying the authenticity of the model after interference or lack thereof. Each of the above steps is described in detail in the following sections.

Determining places to hide watermarks

A fundamental aspect in the domain of watermarking pertains to the imperceptibility of the embedded watermark, encompassing both the visual aspect and the overall distortion of the model. In order to enhance the level of imperceptibility, a novel algorithm, designated as the high-density mesh locator (HDML), has been developed. In regions characterised by dense geometry,

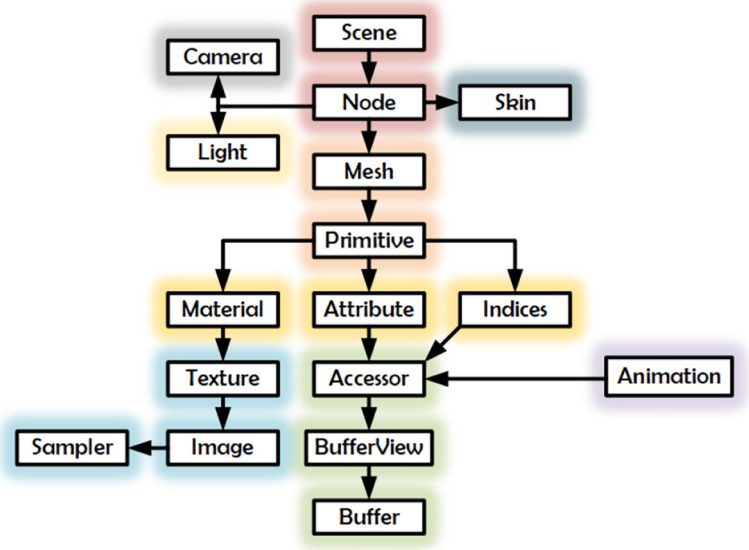


Figure 1. The relationship between the concepts used in glTF 2.0

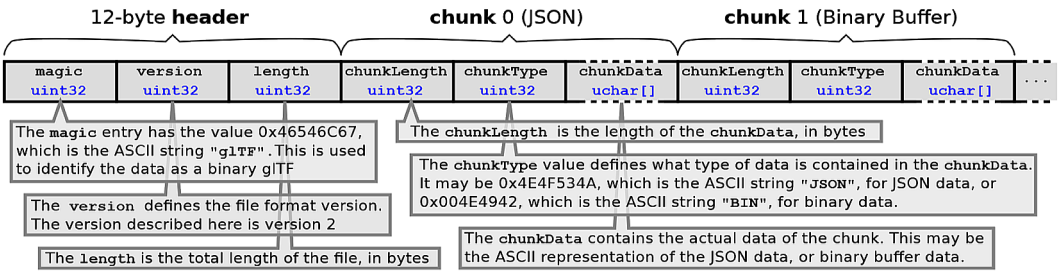
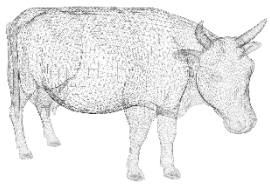
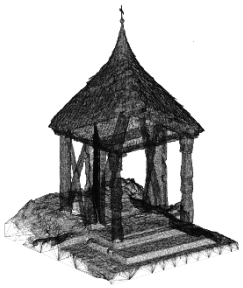
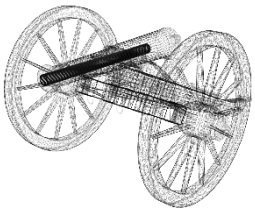


Figure 2. The organizational framework of the .glb file format. Retrieved from https://www.khronos.org/files/gltf20-reference-guide.pdf

Table 1. The three-dimensional assets utilized throughout this study

Model	Parameters	Model	Parameters
	Name: cow		Name: troll
	Size: 18’151KB		Size: 25’438KB
	Vertices: 15’582		Vertices: 48’624
	Triangles: 29’345		Triangles: 81’831
	Extras: texcoord, normals, tangents		Extras: texcoord, normals, tangents
	Source: computer design		Source: computer design
	Name: shoe		Name: gate
	Size: 29’887KB		Size: 25’030KB
	Vertices: 74’321		Vertices: 362’771
	Triangles: 144’527		Triangles: 499’976
	Extras: texcoord, normals		Extras: texcoord
	Source: photogrammetry		Source: scan 3D
	Name: cannon		Name: torch
	Size: 1’245KB		Size: 59’616KB
	Vertices: 63’461		Vertices: 2’996
	Triangles: 84’994		Triangles: 4’260
	Extras: -------		Extras: texcoord, normals
	Source: computer design		Source: computer design

distortions become less perceptible, making them optimal locations for embedding watermarks [53,54]. The proposed algorithm is founded upon the *kd*-tree data structure [55], a binary spatial tree employed for the organisation of points in *k*-dimensional space. This feature facilitates the execution of efficient spatial queries, including the search for the nearest neighbour.

The initial phase of the HDML algorithm entails zeroing the least significant bit (LSB) in each coordinate of each position vector included in the model. HDML operates on a copy of the model. The action is dictated by the necessity of ensuring an identical *3d*-tree structure ($k = 3$) during the verification process. A components of the watermark are integrated into the LSB of the vertex position coordinates, thereby influencing the manner in which the spatial tree is structured. In the subsequent step, a *3d*-tree structure is created, where the input data are vertex position vectors (hereinafter referred to as points). Then, for each point, its nearest neighbourhood is determined, consisting of q points, where q is a parameter of the algorithm. The determination of the neighbourhood is followed by the calculation of the total distance (the l^2 metric) from a given point to all of its neighbours. Points for which the sum of distances is zero are rejected, as this represents a degenerate case in which the points overlap. The algorithm ends by returning m indices of vertices with the smallest total distance, where m is an algorithm parameter that depends on the required space for hiding the watermark bits. Listing 1 summarises the HDML algorithm.

The overall complexity of the algorithm, in terms of Big O notation, is $O(q \cdot n \cdot \log n)$. Step 4 (zeroing the LSB) entails iterating over all vertices. However, each vertex consists of attributes that have a constant shape. For example, a position attribute always contains 3 values, and a texture coordinate always contains 2 values. Therefore, the complexity of step 4 is $O(n)$.

Standard time to build a balanced spatial tree (like a kd tree) is $O(n \cdot \log n)$, and the typical complexity of a single search query is $O(n$

· log n), since a kd-tree is a binary tree. Step 6 specifies *identify for each vertex its nearest q-neighbours using the 3d-tree*. Therefore, the algorithm must perform $n \cdot q$ search operations, making the overall complexity of step 6 $O(q \cdot n \cdot \log n)$. If a standard list were used in place of a kd-tree, the computational complexity would be $O(n^2)$.

Steps 7–9 encompass only simple arithmetic operations and data movement performed for each vertex. The complexity of each of these steps is $O(n)$, since the operations must be applied to every vertex. Subsequently, the sorting operation has a complexity of $O(n \cdot \log n)$ when an efficient algorithm, such as QuickSort, is used.

That is why the overall computational complexity of the HDML algorithm can be expressed as $O(q \cdot n \cdot \log n)$.

Zeroing the last significant bit

The segmentation stage and the watermark generation stage are preceded by a process of zeroing the least significant bit (LSB) in specific locations in the vertices indicated by HDML. This is because after the watermark is embedded, the geometry undergoes a slight modification, while in the verification process - assuming that the model has not been tampered with - it is necessary to recalculate the HMAC and BLAKE2 hashes from the original, unmodified data.

The total number of LSBs set to zero depends on the number of segments into which the model has been divided and the number of available vertex attributes. Zeroing takes place only within the vertices designated by HDML. For each vertex attribute (e.g., normal vectors and texture coordinates), excluding position, a total of 128 least significant bits (LSBs) in the attribute coordinates are set to zero; see Listing 2.

In the case of the position attribute, the following number of LSBs is set to zero:

$$\Lambda = H + (K + 1) \cdot 128 \tag{2}$$

where: $K + 1$ – number of regions into which the model has been divided + JSON section,
H – HMAC code length in bits.

Segmentation of the mesh

In order to identify areas where modifications have been made, the model is divided into segments using k-means++ clustering [56]. The k-means++ algorithm is essentially used to divide n observations into k clusters, with each observation belonging to only one cluster. Mathematically, the goal of the k-means++ algorithm is to solve the following optimisation problem (Equation 2): for a set of observations $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n)$ from the discrete space $\mathbf{P}$, determine k-sets $\boldsymbol{\Omega} = \{\omega_1, \omega_2, \ldots, \omega_k\}$, where $k \leq n$, based on the expression:

$$\arg\min_{\boldsymbol{\Omega}} \sum_{i=1}^{k} \sum_{\boldsymbol{p} \in \omega_i} \delta(\boldsymbol{p}, \boldsymbol{\mu}_i) \tag{3}$$

where: μ_i is the mean (centroid) of the elements of the set ω_i, and $\delta(\cdot,\cdot)$ is a selected metric measuring distance (usually the l^2 norm).

HIGH-DENSITY MESH LOCATOR ALGORITHM

```
Input: points: a copy of the set of 3D points that create a 3D model
q: number of neighbours
m: number of returned vertices
Output: an array of size m storing the indices of the vertices
Data: M: index array
Function determine_dense_regions(points, q, m):
if points == ∅ then
    return an array of size m that is filled with -1
zero the LSB in each coordinate of all elements in the points array
build a 3d-tree
identify for each vertex its nearest q-neighbours using the 3d-tree
calculate the total distance from a vertex to its q-neighbours
discard vertices for which total distance == 0 or very close to 0
write down the indices of the remaining vertices to array M
sort M in increasing order according to the total distance associated with each vertex.
return M[0:m-1] // return the m first elements in M
```

Listing 1. High-density mesh locator algorithm

```
ZERO LSB ALGORITHM
indices ← HDML(model.vertices)
for attribute in model.vertex_attributes
bit_counter ← 0
  if (attribute.name == "POSITION")
    for v in attribute.data[indices] // Only the selected vertices
      v.x ← zero_lsb(v.x); if (bit_counter == Λ) break
      v.y ← zero_lsb(v.y); if (bit_counter == Λ) break
      v.z ← zero_lsb(v.z); if (bit_counter == Λ) break
  else
    for v in attribute.data[indices] // Only the selected vertices.
      v.x ← zero_lsb(v.x); if (bit_counter == 128) break
      v.y ← zero_lsb(v.y); if (bit_counter == 128) break
      v.z ← zero_lsb(v.z); if (bit_counter == 128) break // if v.z is available
```

Listing 2. Zero LSB algorithm

$$\boldsymbol{\mu}_i = \frac{1}{|\omega_i|} \sum_{\boldsymbol{p} \in \omega_i} \boldsymbol{p}. \tag{4}$$

The *k*-means++ algorithm begins the partitioning process by randomly selecting points that serve as the initial centroids of individual clusters. Next, the centroid refinement phase begins in order to solve the optimisation problem (2). After the model segmentation process is complete, each vertex is marked with the number of the region to which it belongs. The segment information is then sent to the watermark generation stage.

The computational complexity of *k*-means++ is $O(n \cdot k \cdot i)$. It involves processing n vertices, k clusters to be found, and i iterations until the clusters converge. This is the most computationally intensive operation in the method.

Watermark generation

The watermark consists of several parts. The first (main) part of the watermark is a 256-bit HMAC code, calculated on the basis of a model copy, (new .glb file, denoted as *temp.glb* in Figures 3–4), which was created after the LSB zeroing stage. The entire file is used as input to the HMAC algorithm**.** The hash function in HMAC is SHA-256 (SHA-2 family of algorithms), and the key is provided externally. In order to improve resistance against watermark adulteration, authors decided to utilize more than one cryptographic function. Instead of using BLAKE2 throughout the method authors used SHA-256 to compute the HMAC. SHA-2 remains a very secure family of hash functions and is computationally less costly than SHA-3. Next, for each available vertex attribute, a separate 128-bit BLAKE2 hash is calculated based on the elements of that attribute. Next, a BLAKE2 hash is generated for each previously designated model segment, with the input data being a set containing the coordinates of the position of the vertices of the given segment. The JSON section (Figure 2) also receives its own watermark in the form of a BLAKE2 hash.

Watermark insertion algorithm

Model watermarking involves modifying the LSB in the coordinates of vectors that are elements of a given vertex attribute. These LSBs are replaced with bits from the calculated HMAC and BLAKE2 sequences. The replacement only takes place within the vertices designated by the HDML algorithm. The bits of the HMAC calculated for the entire file, the bits of the BLAKE2 hash calculated for the JSON section, and the bits of the BLAKE2 hashes calculated for each segment are embedded in the LSB of the position coordinates. Hashes for other attributes (e.g., normal vectors) are placed in the LSB coordinates of the vectors that make up the attribute. The watermark insertion scheme is shown in Figure 3.

Watermarks extraction, originality verification and tamper localisation

The verification of a model's authenticity is carried out in seven steps: (1) identifying watermark embedding locations; (2) extracting the

watermark (hashes); (3) zeroing the LSB where watermark bits were hidden; (4) segmenting the model; (5) recomputing HMAC and BLAKE2 hashes; (6) comparing the extracted hashes with the calculated ones; (7) confirming the model's authenticity or detecting tampering. For the verification process to work correctly, the same steganographic key used during watermark embedding must be applied. The entire verification workflow is illustrated in Figure 5.

A .glb file can be attacked in various locations. Vertex attributes, the JSON section, and the entire binary buffer are all potential targets for unauthorised modification. The tampering localisation process begins with verifying the watermark applied to the entire file. If the extracted HMAC matches the newly computed HMAC, it indicates that the model is authentic. Otherwise, the procedure for locating unauthorised changes is initiated.

If any of the extracted BLAKE2 hashes related to vertex attributes differ from their recomputed counterparts, the tampered area is identified. The integrity of the JSON section is verified in the same way. Comparing the new and extracted hashes associated with individual segments of the model enables detection of the modified region. If no inconsistencies are found, no action is required, and the next test can proceed or the verification process may be concluded. The order of the tests is arbitrary, as they are independent of one another.

Steganographic key

The steganographic key (stego key) is an integral component of the described method. It contains the information necessary for embedding and extracting the watermark, and therefore should only be accessible to authorised individuals. The full stego key is a byte sequence, where the first two bytes encode the number of neighbours used during the construction of the *3d*-tree structure. The next two bytes indicate the number of segments into which the model is divided. Since the *k*-means++ algorithm begins segmentation by randomly selecting initial points, a seed for the random number generator must be provided. This seed is a number of arbitrary length placed immediately after the segment count. Following the seed is a byte representing the * character, used as a separator.

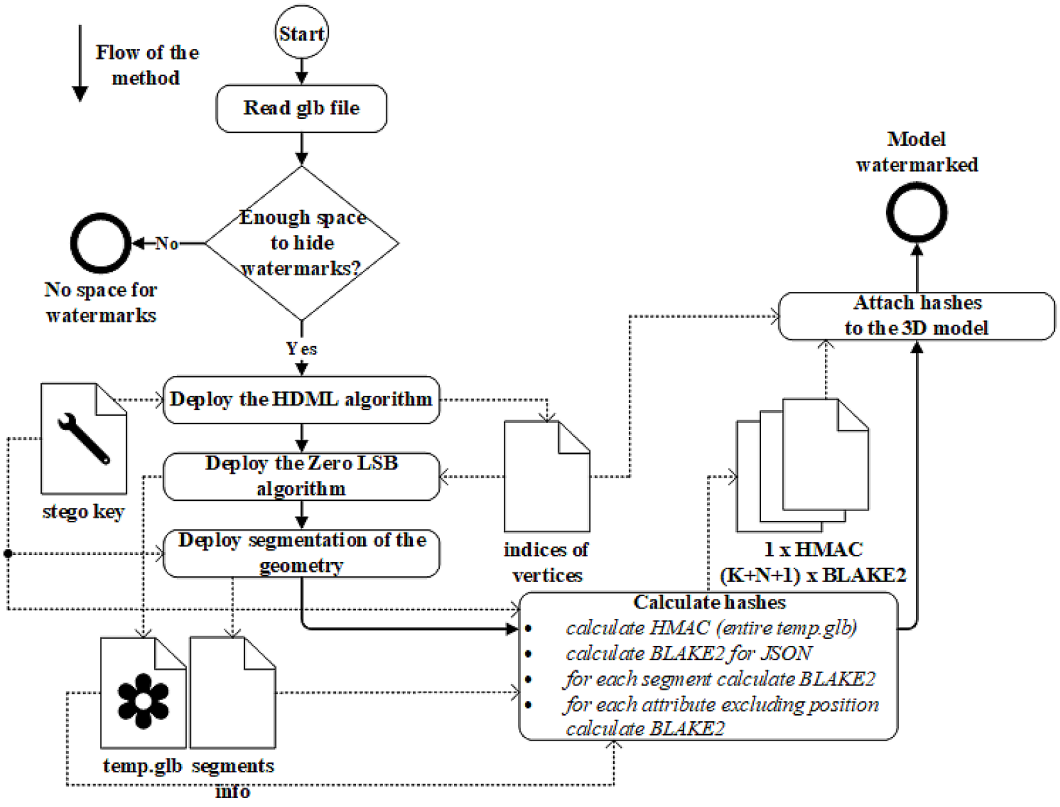


Figure 3. Scheme of the watermarking algorithm. *K* is the number of segments into which the model is divided, *N* is the number of available attributes excluding position

The remainder of the sequence, extending to the end, is the key used for computing the HMAC hash. The structure of the steganographic key is illustrated in Figure 4.

Interference measures

In order to evaluate the proposed method and compare its results with those of other studies, three of the most popular metrics used in the field of 3D watermarking were used.

- Peak signal-to-noise ratio (PSNR) is a measure of the amount of noise, or interference, introduced into the model. PSNR is expressed in dB, and the higher the value of this metric, the better. PSNR is defined as follows [57]:

$$PSNR = 10 \cdot \cdot log_{10} \frac{\left(\max\limits_{\boldsymbol{v} \in \boldsymbol{Y}} \|\boldsymbol{v} - \boldsymbol{c}\| \right)^2}{\frac{1}{card(\boldsymbol{Y})} \cdot \sum_{v \in Y} \|\boldsymbol{v} - \boldsymbol{v}'\|^2} \quad (5)$$

where: **Y** is the set of vertices forming the original model, card(**Y**) is the cardinality of set **Y**, **v**' is a vertex of the modified model corresponding to **v**, and **c** is the centroid of the original model.

$$\boldsymbol{c} = \frac{\left[\sum_{(\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3) \in \boldsymbol{T}} \left(\frac{\boldsymbol{t}_1 + \boldsymbol{t}_2 + \boldsymbol{t}_3}{3} \right) \cdot \right] \cdot \left(\frac{\|(\boldsymbol{t}_2 - \boldsymbol{t}_1) \times (\boldsymbol{t}_3 - \boldsymbol{t}_1)\|}{2} \right)}{\sum_{(\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3) \in \boldsymbol{T}} \left(\frac{\|(\boldsymbol{t}_2 - \boldsymbol{t}_1) \times (\boldsymbol{t}_3 - \boldsymbol{t}_1)\|}{2} \right)} \quad (6)$$

where: **T** is the set of all triangles that form the 3D model. The elements of **T** are tuples containing three vertices (points in $\mathbb{R}^3$) of a triangle.

- To evaluate the total geometric discrepancy between the original and modified models, the root mean squared error (RMSE) is employed, defined by the following equation:

$$RMSE = \sqrt{\frac{\sum_{v \in Y} \|\boldsymbol{v} - \boldsymbol{v}'\|^2}{card(\boldsymbol{Y})}} \quad (7)$$

- To evaluate the fidelity of 3D objects, computer graphics researchers employ the Hausdorff Distance (HD or d_H) metric, which measures the dissimilarity between two different representations of the same 3D object [58].

Given that V_1 and V_2 represent the vertex sets of the altered and initial models, respectively, it follows that:

$$d_H = \max\left(M(V_1, V_2), M(V_2, V_1)\right) \quad (8)$$

where: $M(P_x, P_y) = \max\limits_{\mathbf{p} \in P_x} \mu(\mathbf{p}, P_y)$ is the unilateral distance between two planes P_x, P_y where $\mu(\mathbf{p}, P) = \min\limits_{\mathbf{p}' \in P} \delta(\mathbf{p}, \mathbf{p}')$ is the distance between point **p** and plane P, where $\delta(\cdot,\cdot)$ is the Euclidean distance.

RESULTS AND DISCUSSION

Evaluation of the proposed method

The proposed method embeds an imperceptible, fragile watermark into the model. As a result, any modifications to the file, including single-bit alterations, can be detected. This mechanism reliably verifies the model's integrity and authenticity.

Watermarking every attribute protects against any tampering with vertex attributes. Similarly, the implemented segmentation procedure enables the identification of attacked regions within the geometry.

Figure 6 presents an example of vertex clustering in the model for various numbers of clusters. Each clustering scenario is shown from three different viewpoints. The greater the number of regions into which the model is divided, the more precise the localisation of unauthorised modifications becomes. On the other hand, a higher number of segments requires embedding more watermarks into the model. A potential drawback of this approach is increased distortion of the model. Tables 2–4 present the values of the PSNR,

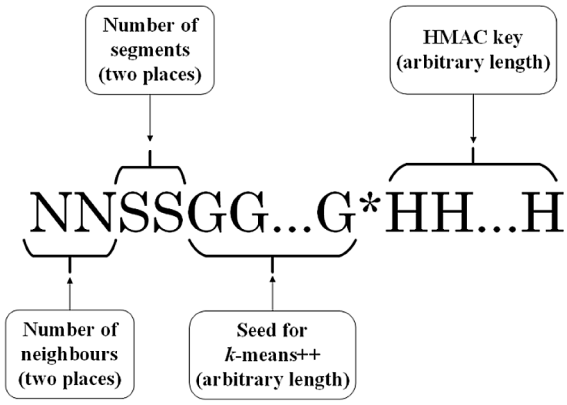


Figure 4. Steganographic key scheme in alphanumerical form

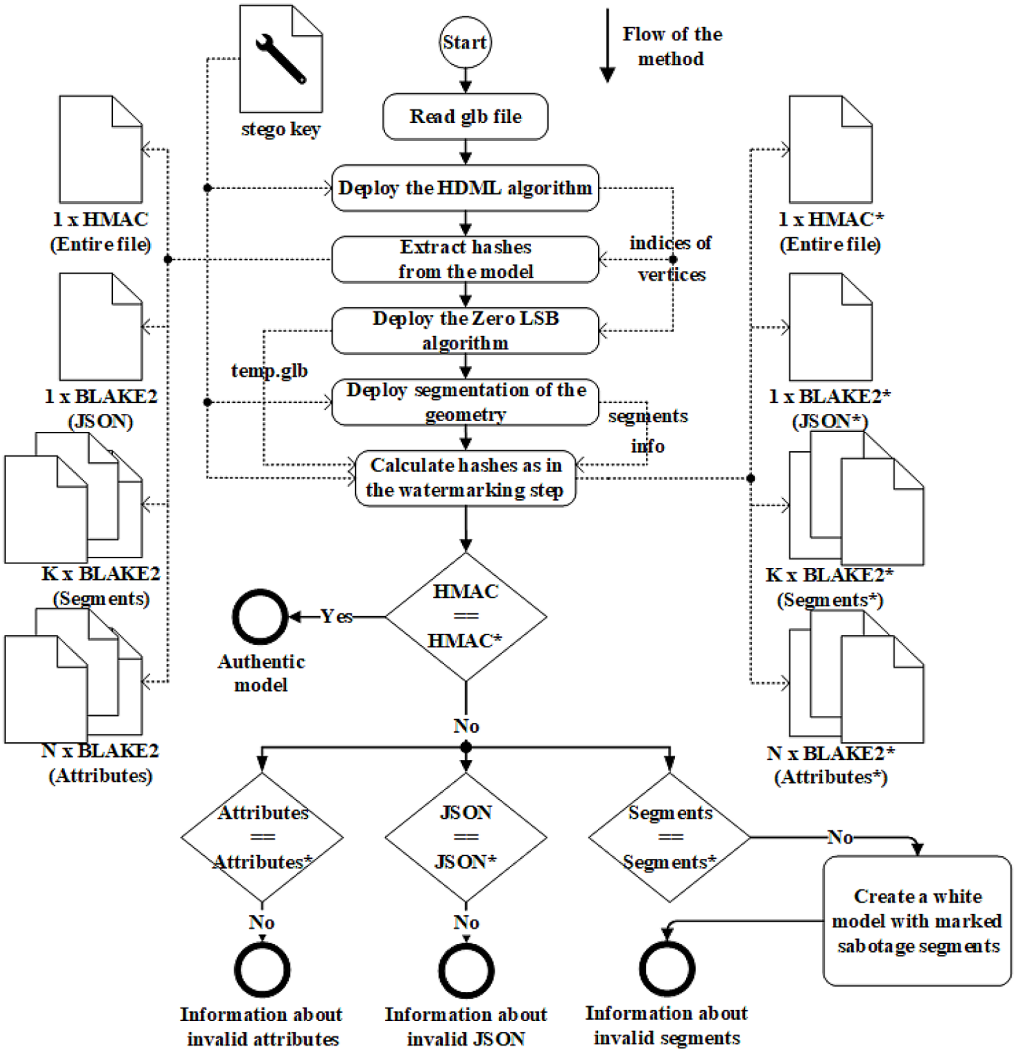


Figure 5. The scheme of the extraction algorithm and verification process. *K* is the number of segments into which the model is divided, *N* is the number of available attributes excluding position

RMSE, and HD metrics, respectively, calculated for different numbers of segments used during the watermarking process, while all other parameters remain consistent across scenarios.

The PSNR values presented in Table 2 clearly confirm that an increased number of watermarked segments leads to higher distortion. However, the PSNR drop is relatively minor – within each row, the difference between the first and last columns is only about 3 to 5 dB. For 20 regions, 2944 bits are embedded into the position attribute. Therefore, dividing and watermarking the model mesh – even for 20 or more segments – is entirely feasible while maintaining a very low level of distortion.

The RMSE metric has favourable results, too. As shown in Table 3, the error value for each model did not exceed the order of 10^{-8}. Changes at this level do not produce any visible artifacts, so the watermarked models appear identical to the originals. The original model and the watermarked model with 20 segments are shown side by side on Figure 7.

To complete the evaluation of the proposed approach, the Hausdorff Distance is utilized as the final performance metric. The results presented in Table 4 confirm the high quality of the developed algorithm, as relatively low HD values were obtained for each model. This indicates that the similarity between the watermarked and original objects, as measured by the HD metric, remains very high.

All three metrics PSNR, RMSE and HD indicate that the method described in this paper embeds the watermark in a way that introduces minimal distortion, thereby achieving the desired level of imperceptibility.

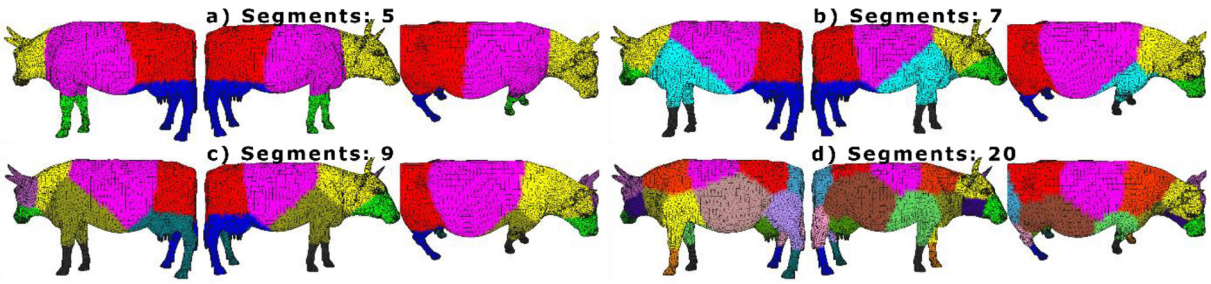


Figure 6. Illustration of segmentation of the model using the *k*-means++ algorithm. a) # segments: 5; (b) # segments: 7; (c) # segments: 9; d) # segments: 20. Each group shows three different snapshots of the model

Table 2. PSNR index for 3D watermarked models with different number of segments. The higher the PSNR value, the better the results

PSNR [dB]				
Name of model	Number of segments			
	5	7	9	20
Gate	231	230	229	226
Shoe	221	220	219	217
Troll	214	213	212	210
Cannon	224	222	222	220
Cow	205	204	204	201
Torch	189	188	187	185

Results of the dense geometry region detection algorithm

A trivial approach to watermark embedding is the random selection of vertices from the entire set. However, this method is not optimal, which is why the authors developed an authorial algorithm high-density mesh locator (HDML) to identify regions with dense and more complex geometry. In areas where the model's mesh is denser – often corresponding to rough or uneven surfaces – embedded information causes less noticeable visual changes. Figure 8 shows the result of the HDML algorithm applied to the *cow* model. A total of 1024 vertex indices were extracted (see Listing 1, parameter m = 1024) and are marked in red.

An analysis of Figure 8 shows that the selected vertices are located in regions with dense – and thus more complex – geometry. Areas such as the corners of the eyes and mouth, nasal cavities, or cloven hooves require a denser mesh for accurate representation compared to, for example, the belly or back. The visible shapes resembling red triangles result from the way the graphics card renders primitives (triangles). Each primitive is defined by a set of vertices, and the regions between them are coloured based on the interpolation of values assigned to those vertices. As a result, we see red triangle-like shapes instead of individual red points.

Discussion of the results of detecting tamper areas

The proposed method enables the detection of attacked regions within a model. During the authenticity verification process, the hash stored in the model for each region is compared with a newly computed hash. A mismatch indicates unauthorised tampering in that region. After the analysis is complete, information about the location of the attack is generated. Several visualisations of this feedback are presented in Figure 9, where the models were divided into different numbers of segments and then subjected to both precise and random modifications of the mesh structure.

Let us consider a scenario in which a third party deliberately attacks the model in order to remove the watermark. To achieve this, the attacker would have to erase (e.g., set to zero) the least significant bits of each vertex attribute. This could be perceived as a drawback, as the operation is relatively easy to perform. However, a model without watermark will fail authentication process. Moreover, there is only one way to

Table 3. RMSE metric for 3D watermarked models with different number of segments. The lower the RMSE value, the better the results

RMSE [-] x10e-8				
Name of model	Number of segments			
	5	7	9	20
Gate	3.76	3.96	4.51	6.31
Shoe	0.07	0.08	0.08	0.11
Troll	0.01	0.01	0.01	0.2
Cannon	0.99	1.26	1.33	1.69
Cow	1.74	1.92	2.00	2.83
Torch	0.70	0.76	0.82	1.11

Figure 7. The original and watermarked models are juxtaposed: a) original models, b) watermarked models. Watermarking was performed using 20 segments

Table 4. HD metric for 3D watermarked models with different number of segments. The lower the HD value, the better the results

HD [-] x10e-8				
Name of model	Number of segments			
	5	7	9	20
Gate	219	219	219	382
Shoe	3	3	3	3
Troll	0.25	0.25	0.25	0.25
Cannon	27	27	27	50
Cow	23	23	23	23
Torch	3	3	3	3

replace the watermark: the attacker would need to obtain access to the steganographic key. Without it, replacing the watermark is not possible as described earlier.

COMPARISON WITH RELATED WORKS

A juxtaposition of watermarking approaches for three-dimensional (3D) models is an intricate and problematic undertaking. The 3D objects utilised in various research projects tend to vary, which complicates the direct comparison of results. Furthermore, the metrics used to evaluate the developed methods are approximate in nature, as there is no single established standard for calculating them. For instance, certain studies opt for the signal-to-noise ratio [54] or the Vertex signal-to-noise ratio [59] as an alternative

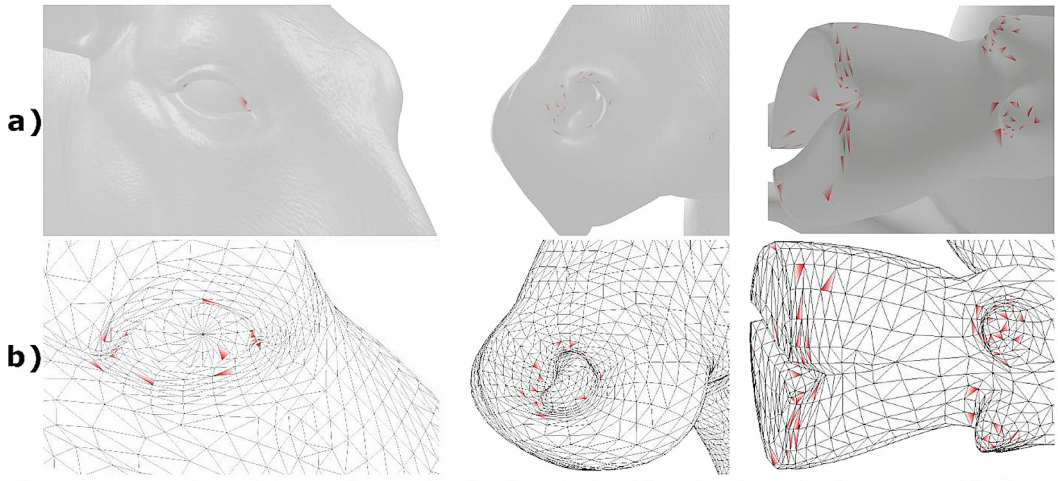


Figure 8. Examples of the results of the developed algorithm for detecting locations with dense geometry. a) solid mode b) wireframe mode

to the PSNR metric. Instead of RMSE, you may encounter $d_{RMSE}(S,S') = \sqrt{\frac{1}{|S|}\cdot\sum_{\mathbf{p}\in S} d(\mathbf{p},S')^2}$ [60]. Therefore, for the purpose of comparison, works containing at least one of the following metrics were selected: PSNR, RMSE, and HD.

In [38] three models were used to evaluate the method, containing 34'835, 23'889 and 428 vertices, respectively. The obtained range of PSNR metric was from ~37 dB to ~40 dB, while the range of HD metric values was from ~2 × 10^{-4} to ~1.25 × 10^{-2}. Our method yielded PSNR values ranging from 185 dB to 226 dB and HD values of the order of 10^{-8}, signifying a substantial enhancement.

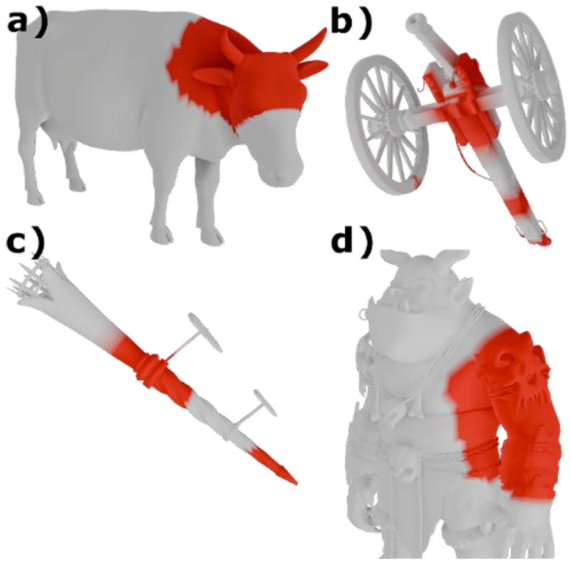


Figure 9. Examples of locating attack areas using the developed method. a) cow model, b) cannon model, c) torch model, d) troll model

The [40] method for a model containing 47'270 triangles obtained a PSNR value of only 44.41 dB; in other cases (less complex models), the result was lower. In the article [32] an RMSE metric value of 10^{-3} was achieved. This result is an order of magnitude (10^{-4} to 10^{-5}) worse than the results obtained with the aforementioned method. Furthermore, the FCM algorithm is more computationally complex than the k-means++ method. The gate.glb model was also used in [37], which allows for a direct comparison of results. Our method yielded PSNR values ranging from 231 dB to 226 dB for the gate.glb model, contingent on the number of segments into which the model was divided. Researches achieved a result of approximately 153 dB in the best case scenario, indicating a discrepancy of 60–70 dB in favour of our method [37].

The article [61] concerns the analysis of methods for hiding information in a VR environment in a distributed manner. Researches conducted an analysis of several scenarios, including one in which they embedded 376 bytes (3008 bits) in a 3D model and obtained a PSNR value of approximately 175 dB. The method employed in our study involved the division of the model into 20 segments, thereby embedding 368 bytes (2'944 bits). This approach yielded PSNR values ranging from 226 dB to 185 dB, which surpasses the results reported by [61].

Additionally, it should be emphasised that none of the presented algorithms consider evaluating the integrity of all the vertices' attributes of

the 3D model. While some authors have alluded to this possibility, none of the cited works address the issue.

CONCLUSIONS

The article presents a novel method for verifying the authenticity of 3D assets in the glTF standard using a fragile watermarking mechanism. To date, relatively little research has addressed watermarking and authentication of models specifically stored in the glTF format. This work therefore provides an in-depth exploration of the glTF standard, extending beyond positional vertices to include all available vertex attributes, such as normal vectors, texture coordinates, and tangent vectors – attributes that are commonly neglected in related studies, which typically focus solely on vertex positions.

In the proposed approach, the watermark is constructed using BLAKE2 and HMAC cryptographic hashes, whose bits are embedded into selected components of the model's vertex data. Importantly, the method is geometry-aware. To determine suitable embedding locations, a new authorial algorithm, termed the HDML, was developed. HDML identifies geometrically complex, irregular regions of the mesh in which even relatively large modifications remain visually imperceptible. This constitutes a significant contribution in its own right. Moreover, HDML is not permanently coupled with the watermarking scheme, which suggests that its applicability may extend beyond the specific use case of watermarking.

Extensive experimental evaluations were conducted on a diverse set of 3D models. Quantitative measures, including PSNR, RMSE, and Hausdorff Distance, were reported to assess the impact of watermark embedding. The obtained results demonstrate that the proposed method reliably detects model modifications while introducing only imperceptible geometric changes, confirming its effectiveness for authenticity verification of glTF assets.

Another potential research direction involves exploring the application of the proposed method to watermarking animated 3D models. Each animation frame could be analysed individually to identify time points where the model's geometry undergoes the greatest deformations and distortions.

REFERENCES

1. Evangeline S. Cybersecurity in Remote Work. In: Yadav M, Pandey A, Huzooree G, editors. Global Work Arrangements and Outsourcing in the Age of AI, IGI Global Scientific Publishing; 2025, p. 227–40. https://doi.org/10.4018/979-8-3373-1270-5.ch012
2. Kuzior A, Tiutiunyk I, Zielińska A, Kelemen R. Cybersecurity and cybercrime: Current trends and threats. Journal of International Studies 2024;17(2):220–39. https://doi.org/10.14254/2071-8330.2024/17-2/12
3. Koziel G, Dziuba-Koziel M. The importance of computer systems security course in computer science studies curricula – A case study. EDULEARN17 Proceedings, IATED; 2017, p. 1227–34. https://doi.org/10.21125/edulearn.2017.1260.
4. Derkach A. History of the development and current state of the 3D modelling. Scientific Bulletin of South Ukrainian National Pedagogical University Named after K D Ushynsky 2023:7–14. https://doi.org/10.24195/2617-6688-2023-4-1
5. Montusiewicz J, Barsz M, Dziedzic K. Photorealistic 3D digital reconstruction of a clay pitcher. Advances in Science and Technology Research Journal 2019;13(4):255–63. https://doi.org/10.12913/22998624/113276
6. Barszcz M, Dziedzic K, Skublewska-Paszkowska M, Powroźnik P. 3D scanning digital models for virtual museums. Comput Animat Virtual Worlds 2023;34(3):e2154. https://doi.org/10.1002/cav.2154
7. Bartol K, Bojanić D, Petković T, Pribanić T. A review of body measurement using 3D scanning. IEEE Access 2021;9:67281–301. https://doi.org/10.1109/ACCESS.2021.3076595
8. Haleem A, Javaid Mohd. 3D scanning applications in medical field: A literature-based review. Clin Epidemiol Glob Health 2019;7(2):199–210. https://doi.org/10.1016/j.cegh.2018.05.006
9. Maneli M, Isafiade O. 3D Forensic crime scene reconstruction involving immersive technology: A systematic literature review. IEEE Access 2022;10:88821–57. https://doi.org/10.1109/ACCESS.2022.3199437
10. Skublewska-Paszkowska M, Powroźnik P, Łukasik E, Smołka J. tennis patterns recognition based on a novel tennis dataset – 3DTennisDS. Advances in Science and Technology Research Journal 2024;18(6):159–76. https://doi.org/10.12913/22998624/191264
11. Wach W, Chwaleba K. Bringing rigid bodies to life: Immersion study in motion capture-based virtual realistic animation of vehicle movement. Advances in Science and Technology Research Journal 2026;20(2):15–32. https://doi.org/10.12913/22998624/210133

12. Wach W, Chwaleba K. Gap filling algorithm for motion capture data to create realistic vehicle animation. Applied Computer Science 2024;20(3):17–33. https://doi.org/10.35784/acs-2024-26.
13. Tanwar R, Pilania U, Zamani M, Manaf AA. An analysis of 3D steganography techniques. Electronics (Basel) 2021;10(19). https://doi.org/10.3390/electronics10192357
14. Setiadi DRIM, Ghosal S, Sahu AK. AI-powered steganography: Advances in image, linguistic, and 3D mesh data hiding -a survey. Journal of Future Artificial Intelligence and Technologies 2025;2:1–23. https://doi.org/10.62411/faith.3048-3719-76
15. Kozieł G. Fourier transform based methods in sound steganography. Actual Problems of Economics 2011;120:321–8.
16. Laftah M, Hamid I, Ali N. Video copyright protection. International Journal of Interactive Mobile Technologies (IJIM) 2023;17(8):135–45. https://doi.org/10.3991/ijim.v17i08.39339
17. Rai M, Kesarwani A. Image watermarking in the digital era: A review of transform, ML-based, and evolutionary methods. Circuits Syst Signal Process 2025. https://doi.org/10.1007/s00034-025-03437-7
18. Zhu J, Zhang Y, Zhang X, Cao X. Gaussian model for 3D mesh steganography. IEEE Signal Process Lett 2021;28:1729–33. https://doi.org/10.1109/LSP.2021.3107777
19. Mabrouk KM, Semary NA, Abdul-Kader H. Fragile Watermarking Techniques for 3D Model Authentication: Review. In: Hassanien AE, Azar AT, Gaber T, Bhatnagar R, F. Tolba M, editors. The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019), Springer International Publishing; 2020, p. 669–79. https://doi.org/doi.org/10.1007/978-3-030-14118-9_66
20. Daiyrbayeva E, Merzlyakova E, Yerimbetova A, Mukhitova A. Reversible Steganographic System for the Transmission of Personal Medical Data. 2024 9th International Conference on Computer Science and Engineering (UBMK), Antalya: IEEE; 2024, p. 1–6. https://doi.org/10.1109/UBMK63289.2024.10773542
21. Daiyrbayeva E, Yerimbetova A, Merzlyakova E, Sadyk U, Sarina A, Taichik Z, et al. An adaptive steganographic method for reversible information embedding in x-ray images. Computers 2025;14(9). https://doi.org/10.3390/computers14090386
22. Matczuk M, Kozieł G, Cięszczyk S. A fragile watermarking scheme for authenticity verification of 3D models in GLB format. Applied Sciences 2025;15(13):7246. https://doi.org/10.3390/app15137246
23. Gorbal M, Shelke R, Joshi M. A Review on Digital Image Watermarking Techniques Using Neural Networks. In: Choudrie J, Mahalle PN, Perumal T, Josh i A, editors. IOT with Smart Systems, Singapore: Springer Nature Singapore; 2023, p. 613–23. https://doi.org/10.1007/978-981-99-3761-5_54
24. Daiyrbayeva E, Yerimbetova A, Maratov Z, Sakenov B. Information Hiding in Images Using Neural Network. 2023 8th International Conference on Computer Science and Engineering (UBMK), Burdur: IEEE; 2023, p. 444–8. https://doi.org/10.1109/UBMK59864.2023.10286608
25. Ben Salah R, Zaied M. Beta wavelet neural networks for medical image watermarking: A fast and robust approach. International Journal of Online and Biomedical Engineering (IJOE) 2025;21(10):94–108. https://doi.org/10.3991/ijoe.v21i10.56201
26. Pal P, Ghosh S, Biswas P, Kar N, Sarkar JL. Secured Digital Watermarking Using Neural Networks. In: Kumar Sahu A, editor. Multimedia Watermarking: Latest Developments and Trends, Singapore: Springer Nature Singapore; 2024, p. 49–66. https://doi.org/10.1007/978-981-99-9803-6_3
27. Botta M, Cavagnino D, Esposito R. NeuNAC: A novel fragile watermarking algorithm for integrity protection of neural networks. Inf Sci (N Y) 2021;576:228–41. https://doi.org/10.1016/j.ins.2021.06.073
28. Uchida Y, Nagai Y, Sakazawa S, Satoh S. Embedding Watermarks into Deep Neural Networks. Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ACM; 2017, p. 269–77. https://doi.org/10.1145/3078971.3078974
29. Nan R, Zhang L, Jin Y, Xie J, Liu S, Wang H. Robust watermarking algorithm for oblique photography 3D models based on multi-level curvature and weighted distance. Earth Sci Inform 2025;18(3):321. https://doi.org/10.1007/s12145-025-01836-7
30. Jiao Y, Ma C, Luo J, Qiu Y. Security protection of 3D models of oblique photography by digital watermarking and data encryption. Applied Sciences 2023;13(24). https://doi.org/10.3390/app132413088
31. Soliman MM, Hassanien AE, Onsi HM. A blind 3D watermarking approach for 3D mesh using clustering based methods. International Journal of Computer Vision and Image Processing (IJCVIP) 2013;3(2):43–53. https://doi.org/10.4018/ijcvip.2013040104
32. El Zein O, M. El Bakrawy L, Ghali N. A robust 3D mesh watermarking algorithm utilizing fuzzy c-means clustering. Future Computing and Informatics Journal 2017;2(2):148–56. https://doi.org/10.1016/j.fcij.2017.10.007
33. Matczuk M, Sulowska D, Gromaszek K. On the usability of isolation forest for 3D mesh analysis and watermarking. Applied Sciences 2025;15(21). https://doi.org/10.3390/app152111364
34. Kesarwani A, Khilar PM. Development of trust

based access control models using fuzzy logic in cloud computing. Journal of King Saud University - Computer and Information Sciences 2022;34(5):1958–67. https://doi.org/10.1016/j.jksuci.2019.11.001

35. Skublewska-Paszkowska M, Powroźnik P, Karczmarek P, Łukasik E, Smolka J. Fuzzy c-means clustering for motion capture tennis time-series data. IEEE Access 2024;12:150975–96. https://doi.org/10.1109/ACCESS.2024.3463201
36. El Zein O, M. El Bakrawy L, Ghali N. A non-blind robust watermarking approach for 3D mesh models. J Theor Appl Inf Technol 2016;83(3):353–9.
37. Kozieł G, Malomuzh L. 3D model fragile watermarking scheme for authenticity verification. Advances in Science and Technology Research Journal 2024;18:351–65. https://doi.org/10.12913/22998624/194146
38. Laftah M. 3D model watermarking based on wavelet transform. Iraqi Journal of Science 2021;62(12):4999–5007. https://doi.org/10.24996/ijs.2021.62.12.36
39. Elhamzi W, Jallouli M, Bouteraa Y. High efficiency crypto-watermarking system based on Clifford-multiwavelet for 3D meshes security. Computers, Materials & Continua 2022;73(2):4329–47. https://doi.org/10.32604/cmc.2022.030954
40. Ali N. Data hiding in 3D model based on surface properties. Iraqi Journal of Science 2023;64(2):973–83. https://doi.org/10.24996/ijs.2023.64.2.39
41. Yerimbetova A, Daiyrbayeva E, Merzlyakova E, Fionov A, Baisholan N, Turdalyuly M, et al. Creating digital watermarks in bitmap images using LaGrange interpolation and Bezier curves. J Imaging 2023;9(10). https://doi.org/10.3390/jimaging9100206
42. Daiyrbayeva E, Yerimbetova A, Nechta I, Merzlyakova E, Toigozhinova A, Turganbayev A. A study of the information embedding method into raster image based on interpolation. J Imaging 2022;8(10). https://doi.org/10.3390/jimaging8100288
43. Al-Qudsy ZN, Shaker SH, Saeed NA. Hashed Watermarking Algorithm for 3D Authentication Model. 2nd Scientific Conference of Computer Sciences, 2019, p. 69–74. https://doi.org/10.1109/SCCS.2019.8852595
44. Lavoué G, Gelasca ED, Dupont F, Baskurt A, Ebrahimi T. Perceptually driven 3D distance metrics with application to watermarking. In: Tescher AG, editor. Applications of Digital Image Processing XXIX, vol. 6312, SPIE; 2006, p. 63120L. https://doi.org/10.1117/12.686964
45. Cohen-Steiner D, Morvan J-M. Restricted Delaunay triangulations and normal cycle. Proceedings of the Nineteenth Annual Symposium on Computational Geometry, Association for Computing Machinery; 2003, p. 312–21. https://doi.org/10.1145/777792.777839
46. Kęsik J, Żyła K, Montusiewicz J, Miłosz M, Neamtu C, Juszczyk M. A methodical approach to 3D scanning of heritage objects being under continuous display. Applied Sciences 2023;13(1):441–61. https://doi.org/10.3390/app13010441
47. Kwieciński K, Paluch M, Lisowska A. Analysis of the dimensional accuracy of point clouds created by photographic scanning. Advances in Science and Technology Research Journal 2024;18(6):121–32. https://doi.org/10.12913/22998624/191430
48. Ghaeedi A, Noroozi E. Development and implementation of hash function for generating hashed message. Advances in Science and Technology Research Journal 2016;10(31):31–9. https://doi.org/10.12913/22998624/64016
49. Tuiri SE, Sabil N, Benamar N, Kerrache CA, Kozieł G. An EEG Based Key Generation Cryptosystem using Diffie-Hellman And AES. 2019 2nd IEEE Middle East and North Africa Communications Conference (Ieeemenacomm'19), 2019, p. 54–9. https://doi.org/10.1109/menacomm46666.2019.8988578
50. Bidakhmet Z, Ziyatbekova G, Rysbayeva A, Darkenbayev D, Mekebayev N, Kyzaibek K. Studying the security of web services USING JSON web Token with RSA-512 algorithm and comparative analysis of HMAC-HA512, HMAC-256 AND RSA-512 for signing Json web token. Bulletin of the National Engineering Academy of the Republic of Kazakhstan 2024;93:42–52. https://doi.org/10.47533/2024.1606-146X.48
51. Roomi G. Improving and protecting the privacy of data security using Blake2 algorithm. Journal of Information Systems Engineering and Management 2025;10:281–93. https://doi.org/10.52783/jisem.v10i5s.625
52. Aumasson J-P, Neves S, Wilcox-O'Hearn Z, Winnerlein C. BLAKE2: Simpler, Smaller, Fast as MD5. In: Jacobson M, Locasto M, Mohassel P, Safavi-Naini R, editors. Applied Cryptography and Network Security, Springer Berlin Heidelberg; 2013, p. 119–35. https://doi.org/10.1007/978-3-642-38980-1_8
53. Taubin G. Geometric signal processing on polygonal meshes. Eurographics State of the Art Reports 2001;4:81–96.
54. Wang Y, Liu J, Yang Y, Ma D, Liu R. 3D model watermarking algorithm robust to geometric attacks. IET Image Process 2017;11(10):822–32. https://doi.org/10.1049/iet-ipr.2016.0927
55. Tiwari V. Developments in KD Tree and KNN Searches. Int J Comput Appl 2023;185(17):17–23. https://doi.org/10.5120/ijca2023922879
56. Arthur D, Vassilvitskii S. k-means++: the advantages of careful seeding. Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete

Algorithms, Society for Industrial and Applied Mathematics; 2007, p. 1027–35.

57. Botta M, Cavagnino D, Gribaudo M, Piazzolla P. Fragile watermarking of 3D models in a transformed domain. Applied Sciences 2020;10:3244–59. https://doi.org/10.3390/app10093244
58. Medimegh N, Belaid S, Atri M, Werghi N. 3D mesh watermarking using salient points. Multimed Tools Appl 2018;77(1):32287–309. https://doi.org/10.1007/s11042-018-6252-6
59. Soliman MM, Hassanien AE, Onsi HM. Robust watermarking approach for 3D triangular mesh using self organization map. 8th International Conference on Computer Engineering & Systems, 2013, p. 99–104. https://doi.org/10.1109/ICCES.2013.6707181
60. Basyoni L, Saleh H, Abdelhalim M. Enhanced Watermarking Scheme for 3D Mesh Models, The 7th International Conference on Information Technology; 2015, p. 612–9. https://doi.org/10.15849/icit.2015.0107
61. Szymczyk T, Czajka P. Analysis of the possibility of hiding decomposed information in the virtual reality environment. Advances in Science and Technology Research Journal 2025;19(2):283–95. https://doi.org/10.12913/22998624/195718