Integrity Verification of GLTF 3D Models Using SHA-256 Hashing and Digital Signatures

3D Fragile Watermarking Using SHA-256 and Ed25519

Almer Zain Farisseno - 18224070
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: almerzainfarisseno@gmail.com , 18224070@std.stei.itb.ac.id

***Abstract*—The increasing use of three-dimensional assets in areas such as virtual reality, industrial design, and digital content creation has raised concerns regarding the authenticity and integrity of 3D model files. Although the Graphics Library Transmission Format (glTF) and its binary counterpart (gLB) have become widely adopted standards for efficient 3D asset transmission, mechanisms for verifying their authenticity remain limited. This paper presents a geometry-aware fragile watermarking framework for gLB models that combines SHA-256 cryptographic hashing with Ed25519 digital signatures to provide authenticity verification. The proposed approach employs a dual-path architecture consisting of metadata-based path and geometry-based path. Cryptographic payloads are embedded into the Least Significant Bits of vertices located in high-density mesh regions, while additional verification data are stored within the JSON chunk of the gLB container. Experimental results demonstrate that the proposed method preserves the visual appearance and texture of the original models while maintaining an imperceptible watermark. Robustness evaluation further shows that the framework quite effectively detects unauthorized geometric and metadata modifications, while the dual-path mechanism enables successful verification under metadata-only attacks. These findings indicate that the integration of cryptographic signatures and geometry-aware fragile watermarking provides a solution for ensuring the integrity and authenticity of GLB-based 3D assets.**

Keywords—SHA-256; Ed25519; GLTF; mesh integrity; digital signature; canonical fingerprint; 3D model verification; cryptography; geometric tampering.

I. INTRODUCTION

A. Background

. The The exponential growth of digital communication and the advancement of graphics processing units have driven the widespread integration of three-dimensional (3D) mesh models across numerous professional and entertainment sectors. Today, 3D models are heavily utilized in virtual and augmented reality (VR/AR) ecosystems, the Internet of Things (IoT), industrial manufacturing, cultural heritage preservation, and clinical dataset visualizations [1], [2]. Because modern workflows, especially in the IT industry, increasingly rely on remote collaboration and massive data transfers between geographically dispersed teams, securing the integrity of these 3D assets has become a critical cybersecurity challenge [2]. In high-stakes applications, such as forensic crime scene reconstruction, the physical and legal consequences of utilizing a tampered or unverified 3D model can be catastrophic.

Currently, the Graphics Library Transmission Format (glTF) and its binary counterpart (GLB) have emerged as the dominant standards for rendering and transmitting complex 3D scenes due to their lightweight structure and high efficiency [1], [2]. Despite its immense popularity and vast steganographic potential, the glTF format remains poorly researched in the domain of data watermarking. The digital architecture of these models is highly susceptible to subtle, unauthorized modifications, such as data piracy or authorship theft, that may evade manual inspection. While traditional cryptographic signatures offer a layer of protection, they often limit data access or necessitate the storage and transmission of an additional electronic signature file alongside the primary data [1].

To address these vulnerabilities without the overhead of external signature files, steganographic techniques offer a highly effective mechanism for authenticity verification. Fragile watermarking ensures that even the most minuscule alteration, such as a single bit modification, definitively indicates data tampering [1]. This paper proposes a novel geometry-aware fragile watermarking scheme for glTF 3D models, leveraging cryptographic hash functions (such as SHA-256 and BLAKE2) and digital signatures. By strategically embedding these cryptographic payloads into dense geometric regions, the proposed method achieves a high degree of transparency while maintaining strict integrity protection [2].

II. Literature Review

A. SHA-256

The Secure Hash Algorithm 256-bit (SHA-256) is a deterministic cryptographic hash function standardized by the National Institute of Standards and Technology under the Federal Information Processing Standards Publication 180-4. SHA-256 operates by processing input data in discrete 512-bit message blocks, utilizing a 256-bit internal state initialized with fixed fractional parts of the square roots of the first eight prime numbers.

B. Ed25519

To mathematically bind the geometric hashes to a verified author, the proposed system utilizes the Edwards-Curve Digital Signature Algorithm, specifically the Ed25519 instantiation defined in RFC 8032. Ed25519 is an advanced elliptic curve signature scheme constructed upon the twisted Edwards curve equation.

C. GLTF

Graphics Library Transmission Format (gLTF) is a universal and widely adopted file format and standard designed for the efficient transfer of 3D assets and entire virtual worlds between different applications [1]. This file format can store not just individual models, but complete 3D scenes, including vertices, complex geometry, materials, lighting properties, camera settings, animation data, and transformation matrices.

D. GLB

Graphics Library Binary (gLB) is a binary file format introduced by the gLTF standard specifically to improve real-time loading speeds [1]. Instead of relying on multiple separate files to render a 3D scene, a .gLB file packages all of the content into a single binary buffer. gLB file format itself consist of two chunks, which are JSON chunk and binary buffer chunk. JSON chunk consists of metadata about the 3D model, while binary buffer chunk stores the detailed information about the 3D models e.g. vertices, normal, and texture shading.

E. Watermarking

Watermarking of 3D resources is an advanced steganographic technique that involves attaching a small amount of hidden information directly to a digital object. In the context of 3D models, watermarks are generally categorized into two primary classifications:

- Robust Watermarking

  Robust watermarks are designed to resist various types of modifications and are mainly used for copyright protection.
- Fragile Watermarking

  Fragile watermarks are highly susceptible to any form of interference, making them ideal for verifying data authenticity and detecting unauthorized changes.

III. Design and Implementation

A. Generating Hash and Digital Signatures

To establish authorship and guarantee geometric integrity for three-dimensional (3D) models, a verification framework is implemented utilizing the Ed25519 digital signature scheme. The process begins with the generation of an asymmetric key pair. In this implementation, the keys are generated using the Python `cryptography` library (specifically the `cryptography.hazmat.primitives.asymmetric.ed25519` module). An Ed25519 private key is generated as a cryptographically secure 32-byte seed, from which a corresponding 32-byte public key is mathematically derived via scalar multiplication on Curve25519. The private key remains secure with the author to sign the model, while the public key is distributed or embedded alongside the signature metadata to allow third-party verification.

The framework employs a dual-hash approach to separate strict binary consistency from visual and semantic authenticity. The two hashes required for this mechanism are:

1) Integrity Hash

   The Integrity Hash is designed to verify the exact, bit-level binary preservation of the model's asset files. It is constructed by extracting all mesh primitives from the 3D model, sorting them deterministically by mesh and primitive indices, and serializing their raw vertex attributes (specifically positions, normals, face indices, and texture coordinates).

   These serialized attributes are then hashed using the SHA-256 algorithm. To ensure compatibility with watermarking schemes (such as least significant bit (LSB) embedding), the vertex position coordinates are normalized to mask out the watermark bits prior to hashing, allowing the integrity check to succeed both before and after the watermark is applied.

2) Canonical Hash

   The Canonical Hash is designed to verify visual and semantic consistency, ensuring resilience against benign re-formatting or re-exporting operations. To generate this hash, the vertex positions and normals are quantized to a predefined decimal precision (e.g., four decimal places), and the vertices are lexicographically sorted to establish a deterministic layout independent of memory ordering. The face indices are then re-mapped to align with the new canonical vertex order.

   Finally, the resulting canonicalized structure is serialized and hashed using SHA-256. This representation remains invariant under minor geometric adjustments, vertex re-ordering, or rounding differences introduced by common 3D modeling packages.

To the model, the computed Canonical Hash and Integrity Hash are concatenated into a single message payload (formatted as `canonical_hash||integrity_hash`). This message is signed using the private key to yield a 64-byte Ed25519 digital signature. The resulting signature, the public key, the two hashes, and a generation timestamp are packaged into a metadata bundle embedded within the 3D model. During verification, a recipient computes the current Canonical Hash and Integrity Hash of the model, and then uses the embedded public key to verify the signature against the computed hashes, thereby validating the model's authorship, semantic consistency, and binary integrity.

B. Insertion Method

To optimize the robustness and survivability of the authentication data against common 3D model processing pipelines, the system implements a dual-path verification architecture. This framework embeds verification data in two distinct sections of the GLB container format which are:

1) JSON metadata chunk

   For the metadata-based verification path, the structured JSON chunk of the GLTF file is parsed, and a cryptographic metadata bundle is injected into the root `extras` field. This metadata bundle contains the Canonical Hash, the Integrity Hash, the author's public key, a unique owner identifier, and the Ed25519 digital signature. Because 3D editing tools frequently strip custom metadata during model re-exporting, this metadata path is complemented by a geometry-level verification path embedded directly within the binary buffer chunk.

2) Binary buffer chunk

   For the geometry-level path, a physical watermark payload is embedded into the vertex data within the binary buffer chunk. The High-Density Mesh Locator (HDML) algorithm is first applied to compute vertex density scores across the mesh and identify a seed vertex in the densest region. From this seed, a region-growing algorithm selects a deterministic set of neighboring vertices. The watermark payload, which comprise of a magic identifier, version number, owner ID, integrity hash, random nonce, and cyclic redundancy check (CRC) checksum, is then encoded into the Least Significant Bits of the selected vertices' floating-point coordinates. By utilizing both the JSON metadata chunk for fast cryptographic validation and the binary buffer geometry for robust physical watermarking, the system ensures dual-layer resilience; authorship can still be verified via the extracted geometry watermark even if the metadata chunk is completely stripped or modified.

C. Extraction Method

To evaluate the authenticity and integrity of a 3D model, the verification framework executes a dual-path check consisting of a metadata-based Fast Path and a geometry-based Deep Path. These paths complement each other, offering both efficient validation and resilience against metadata stripping.

1) Fast path

   The Fast Path parses the GLB file's JSON chunk to retrieve the embedded metadata bundle. Once located, the system recomputes the model's current Canonical Hash and Integrity Hash. The verification process then compares the recomputed hashes against the values stored in the metadata. Additionally, the author's public key (retrieved from the metadata) is used to cryptographically verify the Ed25519 signature against the concatenated hashes.

   A successful match across both hashes and a valid signature indicates complete authenticity and geometric integrity, whereas a signature match with a mismatched Integrity Hash indicates that the model's authorship is authentic but the geometry has been re-exported or repackaged.

2) Deep path

   The Deep Path operates independently of the JSON metadata chunk, executing a direct extraction from the geometry stored in the binary buffer. The verification algorithm first normalizes the floating-point vertex coordinates of the mesh by masking their least significant bits. This normalization ensures that the High-Density Mesh Locator (HDML) identifies the identical seed vertex and selects the same set of dense vertices as during the embedding phase.

   The algorithm then extracts the bit sequence from the LSBs of the selected vertices' coordinates to reconstruct the watermark payload. The integrity of the payload is verified using its cyclic redundancy check (CRC) checksum. Upon validation, the embedded integrity hash is compared against the model's recomputed integrity hash, and the owner identifier is extracted to prove ownership, confirming authenticity even if the model's JSON metadata was completely stripped.

IV. RESULTS

A. Noise Results

In any steganographic application, evaluating the imperceptibility of the embedded payload is critical to ensuring the asset's utility remains uncompromised. For 3D models, this is often quantified through noise metrics that measure geometric discrepancies introduced during the watermarking process. By restricting the cryptographic payload insertion to the Least Significant Bits of vertices located exclusively within high-density mesh regions, the proposed algorithm aims to keep visual and structural

distortions well below the threshold of human perception. The subsequent analysis evaluates this imperceptibility using standard geometric distortion metrics before assessing the robustness of the watermark itself.

To qualitatively evaluate this imperceptibility, a side-by-side visual comparison was conducted. The original, unmodified 3D models (displayed on the left) were rendered alongside their watermarked counterparts (displayed on the right) to carefully assess any perceptible alterations. Visual inspection focused heavily on the spatial integrity of the vertices and the fidelity of the mapped textures.

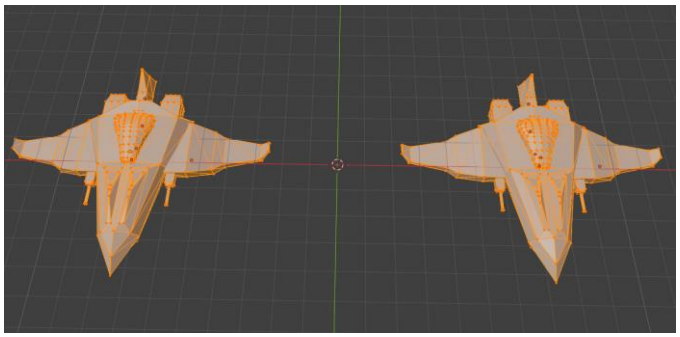


Fig. 1. Comparison of the vertices in original model (left) and watermarked model (right)

The results demonstrate that the code maintained the original model's visual characteristics while introducing no noticeable changes to the vertex positions. This indicates that the code is able to preserve the geometric integrity of the model and did not affect its visual quality.

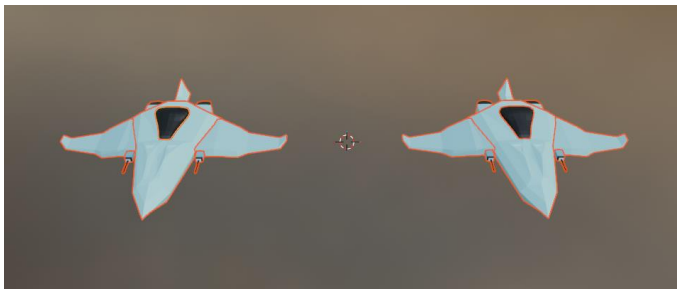


Fig. 2. Comparison of the textures in original model (left) and watermarked model (right)

Based on the comparison results, the code was also able to preserve the texture of the original model. This indicates that the proposed method maintained the visual fidelity of the surface appearance without introducing noticeable texture distortions.

In conclusion, the proposed implementation successfully preserved the appearance of the original file while creating an imperceptible watermark. This demonstrates that the watermarking process can embed information without noticeably affecting the visual quality of the 3D model

B. Watermark Results

To validate the operational effectiveness of the proposed fragile watermarking scheme, the system's sensitivity to unauthorized modifications must be rigorously evaluated. The watermark results assess the algorithm's capability to detect and localize tampering, ranging from minor vertex displacements to structural or metadata alterations.

TABLE I. ROBUSTNESS TEST

No	Tampering Type	Fast Path	Deep Path
1	No Tampering	Pass	Pass
2	Metadata Stripping	Fail	Pass
3	Vertex Modification	Fail	Fail
4	Vertex and Metadat Tampering	Fail	Fail
5	Reexporting	Fail	Fail

Fig. 3. Robustness test table

Based on the table above, the outcomes of the robustness test under various tampering conditions, demonstrating how the dual-path verification framework responds to different forms of data manipulation:

1)Scenario 1 (No Tampering)

The model successfully passes both extraction paths, verifying that the canonical and integrity hashes remain completely intact and readable under normal conditions.

2)Scenario 2 (Metadata Stripping)

The Fast Path fails due to the corruption or removal of the JSON chunk where metadata hashes are stored. However, the Deep Path remains resilient and passes, successfully extracting the watermark directly from the untouched bit chunk.

3)Scenario 3 (Vertex Modification)

Both paths fail. The Fast Path fails because modifying the vertices alters the overall file hash. Simultaneously, the Deep Path fails because the geometric tampering disrupts the canonical hash embedded within the dense mesh regions. This demonstrates the watermark's strict fragility and high sensitivity against any spatial or geometric alterations.

4)Scenario 4 (Vertex and Metadata Tampering)

As expected, simultaneous attacks on both the geometry and the metadata result in a complete failure for both paths, correctly flagging the entire model as heavily compromised.

5)Scenario 5 (Reexporting)

Reexporting a .glb file typically scrambles its binary buffer arrangement and alters the underlying vertex order, even if the visual shape remains the same. Consequently, both paths register a fail, demonstrating

that the hashing mechanisms are highly sensitive to file-level restructuring and effectively detect unauthorized format conversions.

C. Limitations

Since the proposed watermarking scheme means to be a fragile watermarking, it possesses several operational limitations inherent to fragile steganography:

1) Vulnerability to Vertex Re-ordering and Re-exporting

The system exhibits high susceptibility to vertex re-ordering, which commonly occurs when a 3D model is loaded and re-saved by standard modeling tools. During a re-export operation, both verification paths fail. The Fast Path fails because 3D modeling software strips custom JSON metadata by default. The Deep Path fails because the geometry-level watermark extraction is highly dependent on the index-based order of vertices. The seed selection and the neighbor traversal in the region-growing algorithm depend on stable sorting of vertex positions. When the vertex memory layout is shifted or rolled, the extracted bit sequence is completely scrambled, preventing successful parsing and triggering validation failures. Furthermore, since the Integrity Hash is order-dependent, the recomputed integrity hash of the re-ordered mesh will mismatch the original integrity hash stored in the watermark payload, even if the payload itself could be recovered.

2) Sensitivity to Geometric Modifications and Tampering

The framework is extremely sensitive to any spatial modifications made to the model's vertex coordinates. When vertex positions are altered, both verification paths fail. For the Fast Path, the Integrity Hash is a SHA-256 hash computed over the serialized vertex attributes, meaning any binary-level coordinate modification changes the resulting hash entirely and fails the integrity check. For the Deep Path, modifying the vertex coordinates directly alters or overwrites the specific Least Significant Bits where the watermark payload is encoded, leading to a corrupt payload with failed CRC checks. Even if the watermark is successfully parsed, the recomputed integrity hash of the modified geometry will mismatch the original integrity hash embedded in the watermark.

3) Metadata Dependency of the Fast Path

The Fast Path is entirely dependent on the presence of the GLTF `extras` dictionary inside the JSON chunk of the GLB container. In environments where the model is processed through pipelines that prune, sanitize, or optimize the GLTF structure, the custom metadata bundle containing the cryptographic signature and author public key is stripped. In such scenarios, the Fast Path fails immediately and is rendered completely unavailable, forcing the verification pipeline to rely exclusively on the computationally heavier Deep Path.

4) Mesh Complexity Constraints on Watermark Embedding

The geometry-based Deep Path is constrained by the physical complexity and vertex count of the 3D mesh. Because the watermark payload has a fixed binary footprint (76 bytes), it requires a minimum number of vertices to be successfully embedded based on the LSB encoding capacity (e.g., 2 bits per coordinate, providing 6 bits per vertex). If the target 3D model is a low-polygon or highly simplified mesh containing fewer vertices than this required threshold, the watermark cannot be embedded, making the Deep Path entirely unavailable for verifying ownership or integrity.

V. Conclusion

Overall, the study shows that watermarking a .gLTF or .gLB file format can only be done so much, since finding the correct place to insert a robust watermark is a rather difficult task. Although the implemention manage to survive metadata stripping, it wouldn't survive simple modification and re-exporting. Even then, The proposed Implementation contributes to the growing field of secure 3D content management and highlights the potential of combining cryptographic and steganographic techniques for authenticity verification in modern graphics workflows.

Video Link at Youtube

https://youtu.be/Loy2ZV78V9I

Source Code Link at Github

https://github.com/Almerosaurus/3D-Model-Verification

Acknowledgment

First of all, I would like to express my gratitude towards Almighty God for him to make it possible for me to finish this paper within the given deadline. Secondly, I would like to express my deepest gratitude to Mr. Dr. Ir. Rinaldi Munir, M.T. for giving me all the lectures as well as the chance to write this paper. Lastly, I really appreciate all the support that

my family, friends, and teammate on helping to finish this course.

REFERENCES

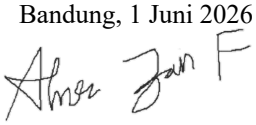
[1] M. Matczuk, G. Kozieł, S. Ci. eszczyk, A Fragile Watermarking Scheme for Authenticity Verification of 3D Models in GLB Format. Appl. Sci. 2025, 15, 7246. https://doi.org/10.3390/app15137246J

[2] M. Matczuk and M. Traczyński, "Geometry-aware fragile watermarking with cryptographic functions for authenticity verification of glTF 3D models," Advances in Science and Technology Research Journal (ASTRJ), vol. 20, no. 7, pp. 139–156, June 2026.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Juni 2026



Almer Zain Farisseno / 18224070