シーングラフ階層を活用した 3Dオブジェクト電子署名と効率的部分検証方式

新井 太陽[1]　大塚 航世[1]　金岡 晃[1,a)]

概要：3D アセットはゲーム、VR/AR、デジタルツインなど多様なサービス間を流通するが、改ざん検知は主にパッケージ単位に限定され、部分的な真正性検証は十分に考慮されていない。本研究は、3D オブジェクトのシーングラフを Merkle 木に写像し、親ノードの電子署名のみで部分木全体を検証可能とする階層ハッシュ署名方式を提案する。提案方式はフォーマット非依存であり、glTF 拡張および USD メタデータとして実装可能である。ユースケースとして Unity ベース VR/AR 環境に組み込み、ビルド時と実行時双方での署名付与と検証を実装した。さらにデジタルツインやロボットシミュレーションへの適用可能性を考察し、本方式がマルチサービス/マルチユーザ/マルチポリシ環境で要求される真正性保証を低コストで提供できることを示す。

キーワード：3D オブジェクト、サプライチェーン、電子署名

Scene-Graph-Aware Digital Signatures for 3D Objects with Efficient Partial Verification

Taiyo Arai[1]　Kousei Otsuka[1]　Akira Kanaoka[1,a)]

Abstract: Three-dimensional assets increasingly traverse heterogeneous services such as games, VR/AR applications, and industrial digital twins, yet integrity protection remains coarse-grained and package-oriented. This paper presents a format-agnostic approach that maps a 3D scene graph onto a Merkle tree, enabling verification of an entire sub-tree by validating only its parent signature. The method embeds hierarchical hash values and X.509-compatible signatures as either a glTF extension or USD metadata, preserving backward compatibility. A case study integrates the scheme into a Unity-based VR/AR workflow: signatures are inserted at build time and dynamically generated or validated at run time. The design is further examined for applicability to digital-twin platforms and robotics simulation pipelines, demonstrating that the proposed mechanism delivers fine-grained authenticity guarantees with logarithmic verification cost in multi-service, multi-user, and multi-policy environments.

Keywords: 3D Object, Supply Chain, Digital Signature

1. はじめに

ゲーム、VR/AR(空間コンピューティング)、産業デジタルツイン、建築 BIM や土木 CAD、ロボットシミュレーション、医療画像、バイオモデリング、3D プリンティングといった多様な分野で 3D のデータが用いられる。そこでは、3D 形状データに加え、そのメッシュ形状やテクスチャ、シェーダなどの付随する情報を含んだ 3D アセットの利用が拡大している。3D アセットは、基本的な図形やメッシュなどの複数のオブジェクトによって構成され、シーングラフに基づく階層構造を形成している。これらのアセットは、複数のサービスやツールを横断して流通・再利用される。3D アセットの価値は高く、セキュリティ上の配慮、とりわけ完全性 (integrity) と真正性 (authenticity) の確保

[1] 東邦大学
Toho University
[a)] akira.kanaoka@is.sci.toho-u.ac.jp

が求められる。本論文では「3D アセット」を、シーングラフ上で参照可能なノードの集合と、それらに結び付くリソース（メッシュ形状、材質やテクスチャ、シェーダ、スケルトン、リグ、アニメーション、物理、衝突属性、スクリプト、メタデータ、外部参照など）を一体として扱う配布・保存・検証の単位と定義する。静的に作成された要素だけでなく、実行時に生成・合成される要素も同じ定義に含める。

「シーングラフ」は、アセット内部の階層関係と参照関係を表すデータ構造である。多くの実装では木構造に近い形で運用されるが、インスタンシングや参照共有により有向非巡回グラフ（DAG）となる場合もある。本論文では、検証の単純化のため、各インスタンスを独立ノードとして扱い、必要に応じて共有リソースへの参照をハッシュ計算に含めることで、根を持つ木として写像して処理する。

データの完全性と真正性は、改ざん検出と出所確認の基礎である。いくつかの分野では、X.509 に基づく電子署名を用いたデータ検証が標準化や運用されている。しかし、適用範囲はパッケージ単位など比較的粗粒度であることが多く、シーングラフ中の個々のオブジェクトや部分木を細粒度に検証する要求に十分対応していない。さらに、空間コンピューティングのように、マルチユーザ/マルチサービス/マルチポリシが同一プラットフォーム上で混在し、アセットの追加や更新、削除、生成が動的かつ頻繁に起こる環境では、従来の静的前提に基づく検証方式は適合しにくい。

VR/AR に代表される空間コンピューティングでは、異種の参加主体がそれぞれのポリシや鍵管理の下でアセットを共有や参照、生成し、実行時にも構成が変化する。この動的かつマルチユーザ/マルチポリシ性においては、各 3D オブジェクトを必要に応じて個別に検証できることが望ましい。一方で、全オブジェクトに電子署名を付与し、実行中に全面的な検証を繰り返す素朴な方法は、署名生成と検証の計算負荷と待ち時間を増大させ、フレーム落ちや遅延を通じて没入感や実在感の低下を招くおそれがある。

本研究は、3D アセットがシーングラフで表現されるという構造的特徴に着目する。具体的には、シーングラフを Merkle 木に写像し、各ノード（オブジェクト）について、子ノードのハッシュ値を含めて自身のハッシュ値を計算し、そのハッシュ値に電子署名を付与する方式を提案する。これにより、あるノードの署名と対応ハッシュを検証するだけで、そのノードを根とする部分木全体の完全性を確認できる。検証は二段階で行う。まず、最上位ノードの署名検証が成功すれば、アセット全体の完全性が担保される。失敗した場合は、疑わしい部分を特定するために子ノード群へ再帰的に降下し、部分木単位で検証を行う。これにより、平均的な検証経路長はシーングラフの深さに比例し、署名検証の回数は深さ程度に抑えられる。

署名付与は開発時と実行時に大別する。開発時には、配布前にビルド成果物へ署名を埋め込む。実行時には、ユーザ生成コンテンツや外部サービスから取り込むアセットに対してオンデマンドに署名を付与する、あるいは受領側で検証を行う。検証は、アプリケーションやサービスの起動時に初期検証を行い、実行中は読み込み、生成、更新、共有といったイベントに応じて対象部分のみを部分検証する。

提案方式はフォーマットやエンジンに依存しない設計であるが、本論文では空間コンピューティングの代表的開発基盤として Unity を取り上げ、ケーススタディとして実装と評価を行う。開発時の署名付与は Unity Editor 拡張として実装し、実行時の署名付与と検証は C#スクリプトとして組み込む。また、開発時と実行時の署名付与、部分検証に要するパフォーマンスを評価する。

2. 関連研究

2.1 3D アセット

本節では、3D アセット内部の階層構造やメタデータ機構を利用する既存の仕様や実装と、それらを用いた検証手法を概観する。代表例を挙げたうえで、本論が目指す「シーングラフ単位の細粒度完全性検証」とどこが一致し、どこが不足しているかを整理する。

2.1.1 3D アセットへの電子署名の適用

Hedberg らは、3D アセットに X.509 証明書を埋め込み、認証や認可、トレーサビリティを可能にする手法を提示している [1]。アセットに署名付きメタデータを保持し、配布後も出所や改変履歴を追えるという特徴がある。製品ライフサイクル全体での信頼確保を想定した用途であり、アセット単位の整合性確認を可能にする。一方で、シーングラフの部分木を対象とする階層的な部分検証や、実行時の軽量な検証といった点は主題にはなっていない。

Acheampong らは、没入型環境での Virtual オブジェクトに対して署名とハッシュを適用する実装を Unity 上に構成し、コントローラボタンといった操作系に署名と検証を結び付ける UI を示した [2]。実装では 2048 ビット RSA と SHA-256 を用い、署名平均 17.3 ms、検証 0–1 ms、追加メモリ約 4 KB といった計測結果を報告し、改ざんやなりすましの検出をシミュレーションで確認した。また、将来の耐量子暗号アルゴリズムへの置換を見据えたモジュール化にも言及している。一方で、対象は基本的にサービス上のアイテム単位の署名と検証であり、シーングラフ階層に沿ったハッシュ構造や、部分検証の経路設計は提示されていない。

2.1.2 フォーマットの拡張とメタデータ付加

実運用で利用されている 3D アセットのフォーマットとして glTF2.0 と USD が知られている。

glTF (GL Transmission Format) 2.0 は拡張メカニズムを備え、署名・ハッシュ・証跡情報を非破壊に付加できる器

として機能する [3]。特に KHR_xmp_json_ld 拡張は XMP に基づくメタデータ格納を標準化しており、出所や利用条件などの情報をアセットに結び付けられるという特徴がある。本研究が焦点としている点と近いが、階層ハッシュ構造の定義、ノード粒度の検証手順、検証時の遅延を抑えるためのプロトコルは仕様上は定めていない。

USD (Universal Scene Description) は参照・レイヤ合成・インスタンシングを中核とし、差分管理や部分差し替えを前提にしたシーン記述を提供する [4]。大規模な合成や非破壊編集を扱いやすいという特徴があり、部分検証という課題設定と狙いが近い。ただし、署名付き階層ハッシュの表現、検証単位の標準的な定義、検証経路の決め方といった点はフォーマット外の設計に委ねられている。

画像・動画分野では C2PA (Coalition for Content Provenance and Authenticity) が、コンテンツに署名付きのマニフェストを付加し、出所と改変履歴を扱う枠組みを提供している [5]。信頼モデルと検証手順が整理され、実運用の事例も増えているという特徴がある。一方で、3D アセットに特有のシーングラフ構造や部分合成、ノード粒度の検証については具体的な規定はない。

2.2 VR/AR セキュリティ

VR/AR 技術とそれにより実現される空間コンピューティングは、3D アセットが実行時に合成・共有・更新される動的な利用形態をとる代表的な領域の一つであり、近年はセキュリティとプライバシの研究関心も高い。

攻撃の系譜はいくつかに分かれる。第一に、センシング基盤や入出力経路を踏まえた従来型に近い手法で、デバイスやセンサから漏れる信号に依存するサイドチャネル型の攻撃である [6], [7], [8], [9], [10]。頭部装着型ディスプレイやコントローラのトラッキングや IMU、内向き/外向きカメラが高頻度かつ高精度のデータを生成するため、推論に利用されやすい。近年は HMD の無権限センサから音声情報を推定する手法など、VR 特有のセンサ構成に着目した研究も報告されている。

従来型の攻撃ではあるが、VR/AR 技術の特徴である 3D 性などから深化する攻撃もある。外向きカメラにより HMD 利用者の周辺にいる人のプライバシーが侵害される可能性 [11], [12] や、Virtual 空間上での中間者攻撃である Man-in-the-Room[13]、Virtual 空間上の覗き見 [14] などが研究されている。また、3D オブジェクトを偽装する攻撃についても報告されている [15], [16]

従来型にはない新たな視点としては、知覚への働きかけを利用する攻撃がある。HumanJoystick のようにユーザの移動や行動を誘導するもの [17]、あるいは PMA (Percptual Manipulation Attack)、VPPM (Virtual-Physical Perceptual Manipulation) の悪用により視覚・聴覚・ハプティクスなど複数感覚にまたがってユーザの課題遂行や判断を乱すものがある [18], [19], [20], [21], [22]。VR/AR コンテンツの設計や提示方法を悪用するか、オーバレイやガイドを差し替えることで、ユーザの知覚や行動に影響を与え、酔いを誘発するという攻撃も指摘されている [23]。

3. システムモデル

3.1 対象と前提

本研究の対象は 3D アセットである。3D アセットは、シーングラフ上のノード集合と、それらに結び付くリソースをひとまとまりの単位として扱う。多くの実装ではインスタンシングや共有参照を持つが、本研究では検証を単純化するために各インスタンスを独立ノードとして扱い、必要に応じて共有リソースの識別子と内容ハッシュを取り込むことで、根を持つ木として処理する。

3.2 参加主体と信頼境界

参加主体は、アセット作成者、配布者、サービス運営者、利用者、第三者の検証者を含む。鍵と証明書の管理は主体ごとに分離される。現状サービスやアプリの多くはマルチユーザ/シングルサービスであるが、空間コンピューティングの進展により、複数サービスが接続され、主体ごとに異なるポリシと鍵管理が併存する状況が想定される。本モデルでは、サービス境界と主体間の鍵境界を明確に区別し、検証の信頼の根は、プラットフォームや組織の認証局（CA）といった各サービスが採用する信頼の基点に置く。

3.3 データフロー

3D アセットのライフサイクルは、作成、パッケージ化、配布、ロード、合成、実行時生成、更新、共有、保存から成る。空間コンピューティングでは実行時合成が頻繁であり、ユーザが持ち込むコンテンツや外部サービスからの参照が逐次追加される。イベントとして、読み込み、生成、更新、共有、破棄を扱う。検証はこれらイベントに結び付けて実施させ、対象部分のみを扱う。

3.4 保証情報

保証情報は、各ノードの内容ハッシュ、子ノードのハッシュ集合から導く親ハッシュ、ノード単位の電子署名で構成する。葉ノードでは内容ハッシュに署名し、内部ノードでは自身の属性と子ノードのハッシュを結合して親ハッシュを計算し、それに署名する。証明書失効情報や鍵の用途制限、ポリシ情報等はメタデータとして併置する。

3.5 署名付与のタイミング

署名付与は開発時と実行時に分ける。開発時は配布前のビルド成果物に対し、ツールで保証情報を付与する。実行時は、ユーザ生成コンテンツや外部サービスから取り込むノードに対してオンデマンドに付与する。実行時署名では

短期鍵やサービス内の委任鍵を用いる構成を想定し、鍵の保護と失効の扱いを明確にする。

3.6 検証の粒度

起動時にルートノードを検証し、整合していればアセット全体の完全性が成立する。整合しない場合は子ノードへ降下し、失敗範囲を部分木として切り出す。実行中はイベントに応じた部分検証を行う。読み込み時は対象部分木を、生成・更新時は影響範囲の親方向にさかのぼって必要最小限のノードを検証する。検証結果と中間ハッシュはキャッシュし、変更の局所性を前提に再計算を抑える。処理は基本的に深さに比例して進むため、全ノードを毎回走査する方法と比べて待ち時間を抑えやすい。

4. 脅威モデル

4.1 想定環境と信頼境界

脅威モデルの対象は、3D アセットが実行時に合成、共有、更新される空間コンピューティング環境である。アプリ/サービスの作成者、配布者、運営者、利用者といった参加主体はそれぞれ独立に鍵とポリシを管理し、シングルサービス内のマルチユーザから、将来的なマルチサービス連携までを含む。信頼の基点は各サービスが採用する基点に置く。署名と検証は完全性と真正性の確認を目的とし、暗号化や利用規約の強制は範囲外とする。

4.2 攻撃者モデル

攻撃者は次のいずれか、または複合で想定する。

(1) アセットストアや User Generated Contents（UGC）といった外部配布チャネルを通じてアセットを投入できる第三者
(2) 正規利用者としてプラットフォームに参加し、自身のアセットをアップロードできる主体
(3) 実行基盤やパーサの脆弱性を悪用できる主体

攻撃者の目的は、アセットの改ざん、差し替え、ノードの不正生成、メタデータの改変、信頼の失墜である。

4.3 攻撃面（Attack Surface）

4.3.1 無害に見せかけた悪意ある 3D アセット

外観は正当だが、内部に不正スクリプトや不正参照を含むアセットが、アセットストアやサプライチェーン経由で混入する経路である。ビルド成果物に取り込まれた場合、実行時には「安全領域内のコンテンツ」と見なされ、ノード生成・属性変更・外部参照の差し替えなどが可能になることがある。

4.3.2 マルチユーザ環境における他者アセットの注入

ネットワーク同期により他ユーザのアセットがローカルにロードされる仕様では、当該アセットの一部が差し替えられていると、ローカルの表示や当たり判定、誘導ガイド等に影響を与える。プラットフォームの同期モデルに依存するが、受領側での検証不備により成立しやすい。

4.3.3 実行基盤・ローダ・パーサの脆弱性悪用

フォーマットパーサの実装欠陥、シリアライゼーションの不備、JIT/ネイティブ連携の境界不備などにより、リモートコード実行（RCE）や任意メモリアクセスに至る経路である。署名と検証のみでは直接阻止できないが、未署名や改変済みのアセットを入口や実行時で遮断することで成立確率を下げることができる。

4.4 動的合成がもたらすリスク

実行中にノードが追加、更新、共有されるため、起動時の一括検証だけでは防御が充分に行えない。ロード、生成、更新、共有といったイベントに応じて、対象部分木のみを低遅延に検証できない場合、上記の攻撃が可視化されず成立しうる。そのため、ノード単位の検証をリアルタイムに繰り返せる仕組みが必要となる。

4.5 防御目標と本方式がカバーする範囲

防御目標は次のとおりとする。

- 完全性：ノード内容と親子関係が改変されていないことを検出する。
- 真正性：署名者（発行者）と内容の結び付けを確認する。
- 局所化：検証失敗時に影響部分木を速やかに切り分け、利用を停止できる。

階層ハッシュとノード署名、そして部分検証を行う本研究の提案方式は、上記目標に直接対応する。

4.6 前提と対象外の要素

暗号プリミティブは標準的な安全性を満たすと仮定する。信頼の基点情報の配布は安全なチャネルで行われると仮定する。可用性攻撃、知的財産保護、レンダリング結果の知覚操作そのものは本方式の直接対象とはしない。これらは別の対策と併用するものとする。

5. 3D オブジェクトのシーングラフに対する効率的な電子署名生成と検証

本章では、シーングラフを下支えにした階層ハッシュ、ノード署名、部分検証からなる方式を提案する。

5.1 データ構造と記法

3D アセットはシーングラフで表し、各ノード v は自身の属性と子ノード集合を持つ。属性はメッシュ・マテリアル・シェーダ参照・外部リソースの内容ハッシュ・物理属性・メタデータ等を正規化したバイト列 $\mathrm{attr}(v)$ とする。子ノード列はシーン内一意 ID の昇順等の安定順序で固定し、$\mathrm{ch}(v)$ で表す。

$$\mathrm{ch}(v) = (c_1, \ldots, c_m), \qquad m \geq 0.$$

暗号学的ハッシュ関数を H とし、ノードハッシュ $h(v)$ は次式で定義する。

$$h(v) = H\big(\mathrm{attr}(v) || h(c_1) || \cdots || h(c_m)\big), \qquad c_i \in \mathrm{ch}(v)$$

葉ノード（$m = 0$）では

$$h(v) = H\big(\mathrm{attr}(v)\big)$$

となる。

各ノードの $h(v)$ に対する電子署名は $\sigma(v) = \mathrm{Sign}_{\mathsf{sk}(v)}(h(v))$ とする（図 1）。外部メッシュなどの共有リソースやコンポーネントなどのオブジェクト関連情報は内容ハッシュと識別子を $\mathrm{attr}(v)$ に含めて同値性を担保する。あるいは、ノードの概念を拡張しオブジェクトだけでなく共有リソースやコンポーネントもノードとして扱い木構造を構成するとすることも可能である。

5.2 署名付与（開発時／実行時）

開発時は配布前のビルド成果物に対し、下位から上位へ $h(\cdot)$ を計算し、各ノードに $\sigma(\cdot)$、証明書鎖、失効参照、ポリシラベル等の検証用メタデータを付与する。実行時は、ユーザ生成コンテンツや外部サービスから取り込むノードに同様の処理をオンデマンドで適用する。いずれもフォーマットの拡張領域に格納し、従来のロード挙動を変えないことを前提とする。

5.3 検証手順

検証は「ハッシュ再計算を伴う署名検証」を基本単位とする。起動時の初期検証では、根ノード r について現在の内容から $h(r)$ を再計算し、次を確認する。

$$\mathrm{Verify}_{\mathsf{pk}(r)}\big(h(r), \sigma(r)\big) = 1 \implies \text{アセット全体の完全性が成立.} \quad (1)$$

失敗した場合は、子 $\mathrm{ch}(r)$ に対して同様の手順を再帰的に適用し、失敗部分木を特定する（図 2）。

$$\mathrm{Verify}_{\mathsf{pk}(r)}\big(h(r), \sigma(r)\big) = 0 \implies \exists c \in \mathrm{ch}(r) : \ \mathrm{Verify}_{\mathsf{pk}(c)}\big(h(c), \sigma(c)\big) = 0 \text{ を探索.} \quad (2)$$

実行時は、読み込み・生成・更新・共有といったイベントに結び付け、影響範囲の部分木のみを検証する。検証結果と中間ハッシュはキャッシュし、不要な再計算を避ける。

5.4 計算量と最適化、現実的な効率

全ノードの再計算・検証は最悪ケースでは $O(n)$ となる。しかし、単一箇所の改変を想定し、根から原因ノードまで

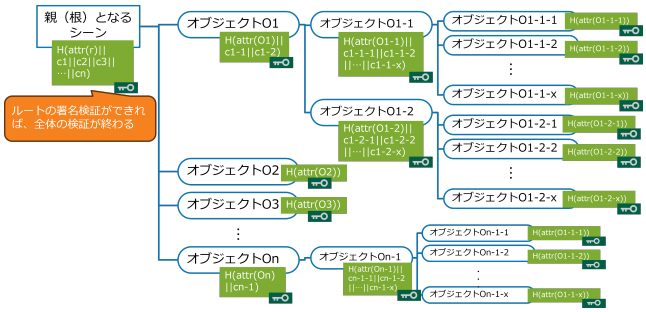

図 1 署名付与のための Merkle 型ハッシュ計算とルートノード検証成功時の概要図

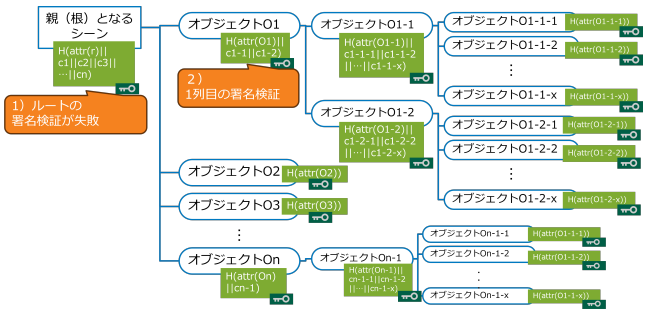

図 2 署名付与のための Merkle 型ハッシュ計算と子ノード検証時の概要図

の経路長を平均深さ d、各階で子ハッシュの集約に要する平均分岐数を k、署名検証のコストは定数項として扱うとすると、必要な再計算は概ね $O(d \cdot k)$ となり効率化が実現できる。木が極端に偏らず、変更が局所的である限り、イベント駆動の部分検証は待ち時間を抑えられる。

また、現実の運用を想定したとき、各オブジェクトの検証失敗は不正なデータの混入が発生していることを示し、その頻度は高くないと考えられる。その場合、検証は根ノード r の検証のみでシーン全体の検証が完了するため、リアルタイム性を担保できると考えられる。

5.5 鍵とポリシ

署名鍵は主体ごとに分離する。開発時は組織鍵やモジュール鍵、実行時のユーザ生成コンテンツにはユーザ本人の鍵以外にもサービス委任鍵や短期鍵を想定する。受領側は信頼する発行者集合と失効の取り扱いをポリシとして明示し、受け入れ可否を判断する。認めない署名は読み込み前に遮断する。

6. Unity を用いた VR/AR 環境での試作

前章で提案したシーングラフに基づく階層ハッシュ、ノード署名、部分検証を Unity で試作実装した。本節では、ビルド時の署名生成、アプリ実行時の署名生成、検証の各機能について述べる。

6.1 ビルド時の署名の生成

Unity Editor の拡張機構を用い、Editor 上のメニューから起動できる署名付与ツールを実装した。

6.1.1 拡張の構成

C#スクリプトにより Editor メニュー項目を追加し、クリック時にダイアログを表示する。ダイアログでは鍵ファイル (PEM) の指定、実行ログの表示を行う。

6.1.2 処理手順

(1) シーンの全ルートから再帰スキャンし、全ノードを収集する。各ノードに固有 ID を付与し、署名メタデータを保持する専用コンポーネントを追加する。

(2) 各ノード v について、前章の定義に従いローカル属性 $\mathrm{attr}(v)$ を正規化し、子ノードのハッシュ $h(c_i)$ と連結して SHA-256 により $h(v)$ を計算する。

(3) $h(v)$ に対して RSA で電子署名 $\sigma(v) = \mathrm{Sign}_{\mathsf{sk}}(h(v))$ を生成し、BASE64 でエンコードしたうえで専用コンポーネントに記録する。

(4) ダイアログに処理結果（対象ノード数、所要時間、各ノードの署名要約）を表示する。

暗号処理は BouncyCastle を用いた。PEM からの鍵読み込み、SHA-256 の計算、RSA 署名と検証を同ライブラリの API で実装している。

6.2 アプリ実行時の署名の生成

実行時に新規生成・外部取り込みされるノードに署名を付与するため、二つの C#スクリプトを用意した。

6.2.1 署名生成スクリプト

入力として対象ノードを受け取り、子ハッシュの集約、SHA-256 による $h(\cdot)$ の計算、RSA による $\sigma(\cdot)$ 生成、専用コンポーネントへの格納までを一つのメソッドで提供する。

6.2.2 実行時トリガスクリプト

テスト検証用に、アタッチされたオブジェクトの生成時 (OnStart()) に上記メソッドを呼び出す。評価では、空の GameObject に本スクリプトを付与し、アプリ起動時にシーン内の全ノードに対して署名付与を行わせた。実運用では、アセットのロード・生成・更新といったイベントに結び付けて必要範囲にのみ署名付与を行う想定である。

6.3 検証

検証は独立の C#スクリプトとして実装した。入力ノードを根とみなし、次の手順で行う。

(1) 入力ノード v の現在の内容から $h(v)$ を再計算し、格納済みの $\sigma(v)$ を公開鍵で検証する。成功なら当該部分木の完全性が成立する。

(2) 失敗した場合、子ノード列 $\mathrm{ch}(v) = (c_1, \ldots, c_m)$ を取得し、各 c_i に対して同手順を再帰的に適用する。これにより失敗原因の部分木を特定する。

本検証器はイベント駆動で呼び出すことを前提とし、読み込みや生成等の各イベントにフック可能である。

表 1 評価結果：小規模シーン（31 オブジェクト）

評価項目	実行時間 (msec)
ビルド時署名生成	229
実行時署名生成	421
公開鍵読み込み	0.36
署名検証（全件）	29

表 2 評価結果：大規模シーン（19,722 オブジェクト）

評価項目	実行時間 (msec)
ビルド時署名生成	142,902
実行時署名生成	194,083
署名検証（全件）	13,343

6.4 パフォーマンス評価

本節では、署名生成（開発時・実行時）および署名検証の処理時間を測定した結果を示す。測定は、基本的なプリミティブ群を用いた小規模シーンと、市街地スケールの大規模シーンの条件で行った。あわせて、オブジェクト数と検証時間の関係について補足測定を行い、線形近似を示す。

6.4.1 評価環境

ビルド（Unity Editor）実行マシンは Windows 11、CPU: Intel Core i7-13700、RAM: 64GB、ストレージ: NVMe SSD（Sequential Read 最大 4950MB/s、Write 最大 4350MB/s）を用いた。アプリ実行デバイスは Meta Quest 3（SoC: Qualcomm Snapdragon XR2 Gen2、RAM: 8GB、内蔵フラッシュストレージ）を用いた。

暗号方式は RSA、鍵サイズは 2048 ビット、ハッシュは SHA-256 を選択した。評価環境は特殊なものではなく、広く普及している一般的な PC と VR デバイス、暗号方式、鍵サイズを用いた。鍵データは PEM 形式テキストからロードした。

6.4.2 ワークロードと手順

小規模シーンでは、カメラやコントローラ等の必須オブジェクトに加え、複数の基本図形を配置し、計 31 オブジェクトに対して測定した。各値は 100 回の実行の平均を求めた。大規模シーンでは、Unity Asset Store の有償アセット「JAPANESE CITY— Modular Pack V1.4」の Prefab シーン Demo を用いた[*1]。オブジェクト総数は 19,722 である。署名検証は提案手法の最悪上限を把握するため全オブジェクトを検証対象とし、部分検証は行っていない。

6.4.3 結果

それぞれの結果を表 1、2 に示す。

6.4.4 補足測定：検証時間の規模依存性

オブジェクト数と検証時間の関係を確認するため、50 オブジェクトと 100 オブジェクトで全件検証を実施した。測定値はそれぞれ 47msec、94msec となった。これらと 31 オ

[*1] https://assetstore.unity.com/packages/3d/environments/japanese-city-modular-pack-v1-4-239043

ブジェクトの小規模シーン環境の結果を合わせ最小二乗の線形近似を求めると、

$$y = 0.9357x + 0.0584, \qquad R^2 = 1.00$$

を得た。x はオブジェクト数、y は検証時間であり、計算には Microsoft Excel を利用した。

6.4.5 考察

RSA の特性上、検証処理は署名処理より軽く、本測定も整合する結果となった。Quest3 の署名検証では、小規模シーンでは 1 オブジェクトあたり約 0.93msec の検証時間という関係が得られた。大規模シーンの全件検証は 13.343sec であり、オブジェクト数で割った平均は約 0.68 となる。シーン構成、I/O、キャッシュの効き方といった測定条件の違いにより単位当たりの時間は変動し得るが、いずれもミリ秒単位である。提案手法は実運用ではルート検証で成功するケースが大半であると考えられ、改変の疑いがある場合にのみ部分木へ降下する。したがって、ここで示した「全件検証」の値は上限評価であり、イベント駆動の部分検証では待ち時間をさらに抑制できる。

7. 他のユースケースでの適用可能性

本章では、Unity で実装をした本提案方式を、他の主要プラットフォームや実行環境に適用可能かを検討する。対象として、Unreal Engine、Apple の ARKit/RealityKit、Google の Android XR、そして WebXR を挙げた。いずれも VR/AR 技術のプラットフォームとして、マルチユーザ／マルチポリシ／マルチサービス化に向かう潮流にあると考えられ、開発時や実行時にアセットが出入りする動的シナリオが想定されうる。

7.1 Unreal Engine への適用

Unreal Engine では、Scene に相当する単位として Level が存在し、その内部に Actor が階層的に配置される。Actor はさらに Component を保持し、機能を追加する仕組みとなっている。この構造はシーングラフとして木構造が Unity と対応関係を持つため、本研究で提案した階層ハッシュと電子署名を利用した検証方式を適用可能である。

具体的には、ビルド時あるいは実行時に Level を根とした Actor 階層をスキャンし、各 Actor に対して署名コンポーネントを追加することで、ハッシュ計算と電子署名の付与を実現できる。Unreal Engine では，C++クラスあるいは Blueprint を用いて新規 Component を実装し、Actor に Add Component することで、署名機能を付与することが可能である。これにより同様の形で、署名および検証の機構を組み込むことができる.

検証処理についても，Actor 生成時のイベントハンドラやアセットロードのフック処理を利用することで、部分木に対するハッシュ再計算と署名検証を行うことができる。

7.2 Apple ARKit/RealityKit への適用

Apple の ARKit や RealityKit においては、Entity と Component を基本単位とした階層構造が採用されている。Entity は位置や回転を含むシーン内のノードとして振る舞い、複数の Component を保持することで、レンダリングや物理演算といった機能を付与できる。これは Unity の GameObject/Component モデルや Unreal Engine の Actor/Component モデルと直接的に対応しており、本研究の提案手法をそのまま適用できる。

開発時には、Xcode のビルド段階で、Scene 全体を走査し、各 Entity に署名コンポーネントを付与することで階層ハッシュを構築できる。実行時には、Entity の生成やロード時にイベントフックを設け、部分木の署名検証を行うことが可能である。Apple プラットフォームでは CryptoKit が標準で利用可能であり、RSA や EdDSA による署名・検証をモバイル性能に最適化して実装できる。

7.3 Google Android XR への適用

Google が提供する Android XR プラットフォームにおいても、Scene Graph は Node と Component を基本とした階層構造を形成する。これにより、Unity や Unreal Engine と同様に、各 Node を単位とした階層ハッシュと署名検証が実装可能である。

開発時には、Gradle ビルドタスクに署名処理を組み込み、配布前にアセットに保証情報を付与できる。実行時には、アセットロード API や OnCreate/OnAttach といったライフサイクルイベントにフックすることで、ロードされた Node に対して部分木の検証を実施できる。鍵管理や証明書流通については、Android Keystore や SecurityConfig を利用できるため、アプリ間やサービス間で異なるポリシを前提とした署名検証をサポートしやすい。

7.4 WebXR への適用

WebXR においては、3D アセットは glTF や USD などのフォーマットで提供され、シーングラフは JavaScript 側で Scene、Object3D といった階層構造で管理される。この構造に対しても、本研究の提案手法は適用可能である。

開発時には、配布する glTF/GLB ファイルに対し、各ノードに署名情報をメタデータとして付与することで、階層ハッシュを組み込むことができる。実行時には、WebXR アプリケーションがアセットをロードする段階で、JavaScript のロード処理にフックし、SubtleCrypto API を用いて署名検証を行うことが可能である。検証に失敗したノードはレンダリングや参照を拒否することで、不正アセットの利用を防ぐことができる。

8. まとめ

本研究では、3D アセットのシーングラフを Merkle 木

に写像し、階層的なハッシュと電子署名による部分検証方式を提案した。Unity での実装評価により、全件検証では処理コストが線形的に増加する一方、部分検証では待ち時間を抑えつつ実用的に動作可能であることを確認し、その実証可能性を示した。また、本方式は Unreal Engine や ARKit/RealityKit、Android XR、WebXR にも適用可能であり、プラットフォームを超えて利用できる汎用性を持つことを示した。今後は暗号方式の拡張、検証経路の最適化、鍵管理やスケーラビリティの評価を進め、セキュリティとユーザ体験の両立を追求する。

謝辞 本研究は、JST、CREST、JPMJCR22M4 の支援を受けたものである

参考文献

[1] Hedberg, Thomas D., J., Krima, S. and Camelio, J. A.: Embedding X.509 Digital Certificates in Three-Dimensional Models for Authentication, Authorization, and Traceability of Product Data, *Journal of Computing and Information Science in Engineering*, Vol. 17, No. 1, p. 011008 (online), DOI: 10.1115/1.4034131 (2016).

[2] Acheampong, R., Popovici, D.-M., Balan, T., Rekeraho, A. and Ramos, M. S.: Enhancing Security and Authenticity in Immersive Environments, *Information*, Vol. 16, No. 3 (online), DOI: 10.3390/info16030191 (2025).

[3] The Khronos 3D Formats Working Group: glTF 2.0 Specification, (online), available from ⟨https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html⟩.

[4] USD Home: Universal Scene Description 25.08 documentation, (online), available from ⟨https://openusd.org/release/index.html⟩.

[5] C2PA: Verifying Media Content Sources, (online), available from ⟨https://c2pa.org/⟩.

[6] Slocum, C., Zhang, Y., Abu-Ghazaleh, N. and Chen, J.: Going through the motions: AR/VR keylogging from user head motions, *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 159–174 (2023).

[7] Luo, S., Hu, X. and Yan, Z.: HoloLogger: Keystroke Inference on Mixed Reality Head Mounted Displays, *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 445–454 (online), DOI: 10.1109/VR51125.2022.00064 (2022).

[8] Meteriz-Yıldıran, Ü., Yıldıran, N. F., Awad, A. and Mohaisen, D.: A Keylogging Inference Attack on Air-Tapping Keyboards in Virtual Environments, *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 765–774 (online), DOI: 10.1109/VR51125.2022.00098 (2022).

[9] Arafat, A. A., Guo, Z. and Awad, A.: VR-Spy: A Side-Channel Attack on Virtual Key-Logging in VR Headsets, *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pp. 564–572 (online), DOI: 10.1109/VR50410.2021.00081 (2021).

[10] Zhang, Y., Slocum, C., Chen, J. and Abu-Ghazaleh, N.: It's all in your head(set): Side-channel attacks on AR/VR systems, *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 3979–3996 (2023).

[11] Denning, T., Dehlawi, Z. and Kohno, T.: In Situ with Bystanders of Augmented Reality Glasses: Perspectives on Recording and Privacy-Mediating Technologies, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, p. 2377–2386 (online), DOI: 10.1145/2556288.2557352 (2014).

[12] Roesner, F., Kohno, T. and Molnar, D.: Security and Privacy for Augmented Reality Systems, *Commun. ACM*, Vol. 57, No. 4, p. 88–96 (online), DOI: 10.1145/2580723.2580730 (2014).

[13] Vondráček, M., Baggili, I., Casey, P. and Mekni, M.: Rise of the Metaverse's Immersive Virtual Reality Malware and the Man-in-the-Room Attack & Defenses, *Computers & Security*, Vol. 127, p. 102923 (オンライン), DOI: https://doi.org/10.1016/j.cose.2022.102923 (2023).

[14] Mathis, F., JosephO'Hagan, Khamis, M., Vaniea, K.: Virtual Reality Observations: Using Virtual Reality to Augment Lab-Based Shoulder Surfing Research, *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 291–300 (online), DOI: 10.1109/VR51125.2022.00048 (2022).

[15] Fujita, M., Kurasaki, S. and Kanaoka, A.: Securing Cross Reality: Unraveling the Risks of 3D Object Disguise on Head Mount Display, *Proceedings of the 13th International Conference on the Internet of Things*, IoT '23, pp. 281–286 (online), DOI: 10.1145/3627050.3631570 (2024).

[16] Fujita, M., Kurasaki, S. and Kanaoka, A.: Investigating 3D Object Spoofing on Fundamental and Custom Objects in Virtual Reality, *HCI for Cybersecurity, Privacy and Trust* (Moallem, A., ed.), pp. 171–187 (2025).

[17] Casey, P., Baggili, I. and Yarramreddy, A.: Immersive Virtual Reality Attacks and the Human Joystick, *IEEE Transactions on Dependable and Secure Computing*, Vol. 18, No. 2, pp. 550–562 (online), DOI: 10.1109/TDSC.2019.2907942 (2021).

[18] Cheng, K., Tian, J. F., Kohno, T. and Roesner, F.: Exploring User Reactions and Mental Models Towards Perceptual Manipulation Attacks in Mixed Reality, *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 911–928 (2023).

[19] Tseng, W.-J., Bonnail, E., McGill, M., Khamis, M., Lecolinet, E., Huron, S. and Gugenheimer, J.: The Dark Side of Perceptual Manipulations in Virtual Reality, *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, (online), DOI: 10.1145/3491102.3517728 (2022).

[20] Kurasaki, S. and Kanaoka, A.: Image Movement Attacks on Optical See-Through HMDs: Covert Gaze Manipulation and Privacy Risks in AR/MR Systems, *2025 IEEE International Conference on Metaverse Computing, Networking, and Applications (MetaCom)* (2025).

[21] Otsuka, K. and Kanaoka, A.: Auditory Stimulus Attack in XR: Stimulus Characteristics and Technical Background Considerations, *2025 IEEE International Conference on Metaverse Computing, Networking, and Applications (MetaCom)* (2025).

[22] Mori, R., Kakizaki, Y., Nakatani, H., Kanaoka, A. and Ohigashi, T.: Attack based on Operational Error caused by Stop-Signal Reaction Time in VR space , *2025 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 881–883 (online), DOI: 10.1109/VRW66409.2025.00179 (2025).

[23] Valluripally, S., Gulhane, A., Hoque, K. A. and Calyam, P.: Modeling and Defense of Social Virtual Reality Attacks Inducing Cybersickness, *IEEE Transactions on Dependable and Secure Computing*, Vol. 19, No. 6, pp. 4127–4144 (online), DOI: 10.1109/TDSC.2021.3121216 (2022).