JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 31 No. 2, pp. 173-181, February 2026
https://doi.org/10.9708/jksci.2026.31.02.173

ERC-7891: A New Protocol Design for Hierarchical NFTs with Splitting and Merging

Nitin Bhagat*, JongWook Bae**, Su-Hyun Lee**

*Student, Dept. of Computer Engineering, Changwon National University, Changwon, Korea
**Professor, Dept. of Computer Engineering, Changwon National University, Changwon, Korea

[Abstract]

This paper presents ERC-7891, a new Ethereum standard that introduces protocol-level mechanisms for splitting, merging and redistributing fractional ownership within a hierarchical NFT (Non-Fungible Token) structure. Existing standards like ERC-721, ERC-1155 and ERC-6150 lacked functionality for dynamic structural changes, limiting their applicability to real-world assets (RWAs) that evolve over time. ERC-7891, proposed in this paper, addresses this gaps by defining standardized functions and events, enabling asset management with transparency and verifiable lineage tracking. The proposed architecture integrates seamlessly with ERC-6150 and ERC-721, incorporating ERC-165 interface detection for interoperability among decentralized applications (DApps). The proposed standard documented the protocol through the formal standardization process, resulting in ERC-7891's successful adoption as an official Ethereum standard. ERC-7891 enables dynamic hierarchical NFTs, providing a foundation for next-generation decentralized applications in domains such as real estate, supply chain management and modular product ecosystems.

▸ **Key words**: Hierarchical NFT, Split and Merge, Ethereum Standard, Blockchain Interoperability

[요 약]

본 논문은 계층적 NFT(대체 불가능 토큰) 구조 내에서 분할, 병합, 부분 소유권 재분배를 위한 프로토콜 수준의 메커니즘을 도입하는 새로운 이더리움 표준인 ERC-7891을 제시한다. ERC-721, ERC-1155, ERC-6150과 같은 기존 표준은 동적 구조 변경에 대한 기능이 부족하여 시간이 지남에 따라 변화하는 실물자산(RWA)에 적용하는 데 한계가 있었다. 본 논문에서 제안하는 ERC-7891은 이러한 격차를 해소하기 위해 표준화된 함수와 이벤트를 정의하며, 이를 통해 투명성과 검증 가능한 계보 추적(lineage tracking)을 통해 자산을 관리할 수 있다. 제안된 아키텍처는 ERC-6150 및 ERC-721과 완벽하게 통합되며, 탈중앙화 애플리케이션(DApps) 간의 상호운용성을 위해 ERC-165 인터페이스 감지 기능을 통합한다. 제안한 표준은 공식적인 표준화 프로세스를 통해 프로토콜을 문서화했으며, 그 결과 ERC-7891은 공식 이더리움 표준으로 성공적으로 채택되었다. ERC-7891은 동적으로 계층적 NFT를 가능하게 함으로써 부동산, 공급망 관리, 모듈형 제품 생태계와 같은 영역에서 차세대 탈중앙화 애플리케이션을 위한 기반을 제공한다.

▸ **주제어**: 계층적 NFT, 분할 및 병합, 이더리움 표준, 블록체인 상호운용성

• First Author: Nitin Bhagat, Corresponding Author: Su-Hyun Lee
*Nitin Bhagat (bhagatnitin312@gmail.com), Dept. of Computer Engineering, Changwon National University
**JongWook Bae (bae@cwnu.ac.kr), Dept. of Computer Engineering, Changwon National University
**Su-Hyun Lee (sleepl@changwon.ac.kr), Dept. of Computer Engineering, Changwon National University
• Received: 2025. 12. 10, Revised: 2026. 02. 03, Accepted: 2026. 02. 12.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

Blockchain technology has transformed asset tokenization by enabling digital and physical assets to be represented on decentralized networks[1]. However, widely adopted NFT standards particularly ERC-721 and ERC-1155 are limited by their flat ownership models, where each token exists as an isolated unit without mechanisms for compositional or hierarchical representation[2, 3]. While ERC-721 popularized unique NFTs for digital artwork, collectibles, and virtual items, it cannot model multi-layered or structurally complex assets.

As blockchain applications expand into areas such as supply chain tracking, modular product representation, real-estate tokenization, and digital property rights, these limitations become increasingly restrictive. For example, in a real-estate tokenization scenario, an entire property can be represented as a parent NFT with full ownership, which is then split into child NFTs representing subdivided units with ownership shares transferred from the parent. This linkage preserves ownership propagation and verifiable lineage across hierarchical levels. Existing NFT standards do not natively support such structured and dynamic asset representation.

ERC-6150 addressed part of the problem by introducing hierarchical NFTs through standardized parent-child relationships, tree-like organization of assets[4]. However, ERC-6150 only supports static hierarchies and lacks protocol-level mechanisms for splitting ownership, merging components, or redistributing fractional shares, which are essential for real-world dynamic asset lifecycles. As a result, existing implementations were forced develop custom extensions, leading to inconsistent behavior and reduced interoperability across platforms[5].

To address these limitations, this paper proposes ERC-7891, a new Ethereum standard that formalizes the dynamic restructuring of hierarchical NFTs through protocol-level support for splitting, merging, fractional share redistribution, and structured token burning. By integrating these mechanisms into the standard itself, ERC-7891 ensures deterministic behavior, cross-platform interoperability, and consistent management of compositional assets, while maintaining hierarchical ownership integrity. This work presents the background and motivation for the standard, its design and architecture, ERC-165 interface detection, and the formal submission and acceptance process of ERC-7891, establishing it as a foundation for next-generation NFTs capable of modeling dynamic, real-world asset (RWA) structures.

II. Background And Motivation

The progression of Ethereum token standards has shaped the evolution of digital asset management. ERC-20 enabled fungible tokens, ERC-721 introduced non-fungible tokens but maintained a flat structure, and ERC-1155 improved efficiency by supporting multiple token types within a single contract[6]. However, none of these standards offered a hierarchical or compositional model suitable for multi-component assets.

Hou-Wan Long et al. introduced ERC-404, a hybrid token model linking ERC-20 and ERC-721 to enable fractionalized ownership and recombination of NFTs. While this approach improved liquidity and composability, it lacked standardized governance for fractional ownership, cross-platform interoperability, and conflict resolution, limiting its practicality and adoption[7].

Bae et al. extended hierarchical asset modeling using Klaytn's KIP-17 standard, enabling parent NFTs to generate child NFTs with assigned ownership shares. Although this structure improved flexibility and recursive asset modeling, the absence of a standardized protocol limited interoperability with other blockchain ecosystems[8].

Bhagat et al. advanced hierarchical NFT

management using ERC-6150, enabling dynamic splitting and merging within a parent-child structure. However, the model lacks a standardized protocol for splitting and merging operations and does not fully address interoperability, limiting its scalability and cross-platform adoption[9].

<Table 1> summarizes the key functional differences among ERC-404, KIP-17, ERC-6150, and the proposed ERC-7891.

Table 1. Comparative Analysis of NFT Standards

Feature	ERC-404	KIP-17	ERC-6150	ERC-7891
Fungible & Non-Fungible	○	×	×	○
Hierarchical Structure	×	×	○	○
NFT Splitting	×	×	×	○
NFT Merging	×	×	×	○
Fractional Ownership	○	×	×	○

Many real-world applications such as supply chain traceability, fractional ownership of high-value items, and modular product assembly, require NFTs that can dynamically restructure as assets evolve. Thus, to address these gap we introduce ERC-7891, a protocol-level standard that formalizes splitting and merging within the hierarchical NFT. By embedding these mechanisms directly into the standard, ERC-7891 provides a unified, interoperable, and consistent framework for managing complex, multi-component, and evolving assets on the Ethereum blockchain.

III. Design and Architecture

The design of ERC-7891 reflects our effort to extend hierarchical NFT capabilities by introducing standardized mechanisms for splitting and merging. To address the limitations of existing ERC standards, this protocol formalizes dynamic operations for hierarchical NFTs. This section presents the technical design principles and architectural structure that guided its development.

1. Conceptual Design

In designing ERC-7891, our primary goal was to define a protocol that integrates splitting and merging directly into the NFT standard rather than leaving such operations as application-level extensions. To achieve this, we introduced a set of standardized functions that govern the full lifecycle of hierarchical NFTs and ensure structural consistency and verifiable share management across all operations. ERC-7891 defines five core functions:

1) mintParent()

This function creates a root token, which serves as the top-level NFT in the hierarchy. The root token is initialized with a full ownership share, forming the foundation on which all subsequent splitting and merging operations depend.

2) mintSplit()

This function divides a parent NFT into one or more child NFTs. The specified share is deducted from the parent and assigned to a new child token. This mechanism ensures proportional ownership distribution while preserving the hierarchical parent-child relationship. Following the technical basis established in our previous work on dynamic hierarchical NFTs, multi-level segmentation is supported, where ownership shares propagate across hierarchical levels, allowing a child's effective share to be derived from its parent's share relative to the grandparent[9].

3) mintMerge()

This function merges multiple child NFTs into a new consolidated NFT. The total ownership share of the selected children is aggregated and carried forward to the newly created token. The original children are burned to maintain structural integrity.

4) sharePass()

This internal redistribution mechanism transfers ownership shares between tokens during splitting, merging, and burning operations. It ensures that share values remain accurate and verifiable throughout the lifecycle of the NFT hierarchy. It supports multi-level share propagation of

ownership, where redistributed shares are consistently adjusted across parent, child, and higher-level ancestors in a hierarchical structure[9].

5) burn()

This function removes a token from circulation. If a child token is burned, its share is reassigned to the parent to maintain ownership continuity. If the burned token is a root NFT, its share is nullified entirely.

2. Contract Architectural Framework

The architecture of ERC-7891 is organized into two main components: the interface specification, IERC7891.sol and the reference implementation, ERC7891.sol. Together, these components establish the functional specification of the protocol and ensure compliance with Ethereum interoperability requirements.

2.1 IERC7891 Interface Specification

As part of the work, we design the IERC7891 interface that defines the core functions and must be implemented by any compliant contract as shown in Algorithm 1. These include mintParent, mintSplit, mintMerge, sharePass, and burn, which together formalize the lifecycle management of hierarchical NFTs.

Algorithm 1. IERC7891 Interface Specification

```
interface IERC7891 is IERC6150
{
    event Split(uint256 indexed parentId, uint256 indexed
                childId, uint8 share);
    event Merged(uint256 indexed newTokenId, uint256[]
                mergedTokenIds);

    function mintParent(string memory _tokenURI) external
                payable returns (uint256 tokenId);
    function mintSplit(uint256 parentId, uint8 _share) external
                payable returns (uint256 tokenId);
    function mintMerge(uint256 parentId, uint256[] memory
                _tokenIds) external payable returns (uint256
                newTokenId);
    function sharePass(uint256 from, uint256 to, uint8 _share)
                external;
    function burn(uint256 tokenId) external;
}
```

Functions are marked as payable where necessary to allow for integration with value transfers during minting or merging operations. In addition, the interface introduces standard events Split and Merged which provide transparency and enable external applications to track structural changes in NFTs directly on-chain. This interface formalizes the key operations of the protocol.

2.2 Reference Implementation

To demonstrate the feasibility and correctness of the proposed standard, we developed ERC7891.sol as the complete reference implementation as shown in Algorithm 2. This implementation operationalizes all core functions defined in IERC7891 and integrates them with the hierarchical management capabilities of ERC-6150. This inheritance structure, illustrated in <Fig. 1>, shows how ERC-7891 integrates with existing Ethereum standards.

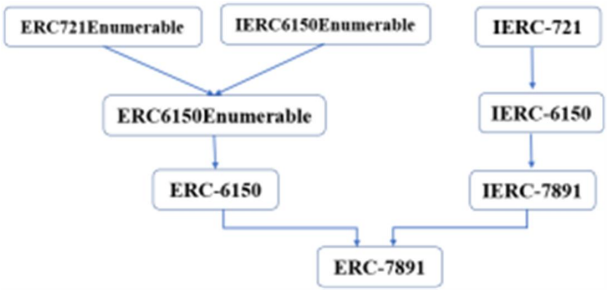

Fig. 1. Inheritance Structure of ERC-7891

As depicted, ERC-7891 inherits functionality from ERC-6150 and ERC6150Enumerable, while simultaneously implementing our custom IERC7891 interface. This ensures that the protocol maintains parent-child relationships, supports enumeration, and remains fully interoperable with tools and systems already built around ERC-721.

Within this architecture, ERC7891.sol introduces dedicated mappings for share management and token metadata, enabling precise accounting of fractional ownership during splitting, merging, and burning operations. The contract also uses

OpenZeppelin's Counters library for secure token ID generation. A simplified representative structure of the implementation is shown below which validates the ERC-7891 protocol.

Algorithm 2. ERC-7891 Reference Implementation

```
contract ERC7891 is ERC6150, IERC7891 {
    Counters.Counter private _tokenIds;
    mapping(uint =>string) public tokenURIs ;
    mapping(uint256 => uint8) public share;

    constructor() ERC7891("ERC7891", "NFT"){    }

    function mintParent(string memory tokenURI) external
                payable override returns (uint256)  {    }
    function mintSplit(uint256 parentId, uint8 _share) public
                payable override returns (uint256) {    }
    function mintMerge(uint256 parentId, uint256[] memory
                tokenIds) public payable override returns
                (uint256) {    }
    function sharePass(uint256 from,  uint256 to, uint8
                _share) public{   }
    function burn(uint256 _tid) public {    }
}
```

The correctness and performance of this reference implementation were validated in our previous work through deployment on a Ganache-based Ethereum test environment[5]. The results showed that the core hierarchical NFT operations execute reliably with acceptable gas consumption and performance.

2.3 Security Considerations

Security considerations were incorporated into the design and implementation of ERC-7891. The reference implementation is based on Solidity ^0.8.0 and mitigates computational vulnerabilities such as integer overflow and underflow. Access control is ensured by verifying token ownership using the _ownerOf() function. Since only legitimate token owners can modify the hierarchical structure, this mechanism prevents unauthorized modifications and incorrect transfers of shares. Furthermore, share transfer operations verify token existence before execution, maintaining consistency within the hierarchy. During burn operations, ownership shares are reassigned to the parent token prior to destruction, preserving ownership integrity.

2.4 Protocol Immutability and Share Invariants

To ensure protocol immutability, ERC-7891 enforces strict share and hierarchical invariants. This invariant can be expressed as:

$$TotalShare = Share(R) + \sum_{i=0}^{n} Share(C_i)$$

Where R denotes the root token and C_i represents its descendant child tokens.

The total ownership share originating from a root token is preserved at all times, preventing artificial inflation or loss of shares. Each token's share is constrained within a valid range and validated during mintSplit() and mintMerge() operations. Furthermore, hierarchical consistency is maintained by ensuring that any change in a child's share is reflected in the parent's remaining share, and parent tokens cannot be burned unless all dependent child shares are properly reassigned or merged.

IV. ERC-165 Interface Detection

ERC-165 provides a standardized mechanism for publishing and detecting the interfaces implemented by an Ethereum smart contract[10]. To ensure full interoperability, with the Ethereum standards, we integrated ERC-165 interface detection into the ERC-7891 standard. ERC-165 provides a universal mechanism that allows external contracts and applications to verify whether a contract implements a particular interface through the function supportsInterface(bytes4 interfaceID). By adopting this mechanism, our standard enables ERC-7891-compliant contracts to be automatically recognized across decentralized applications.

1. Interface ID Computation for IERC7891

As part of this work, we computed the ERC-165 interface identifier for IERC7891 by deriving the selectors of all functions defined in the interface. Each selector was obtained by hashing the function signature using Keccak-256 and extracting the first four bytes. <Table 2> summarizes the computed hashes and selectors.

Table 2. Function Signatures and Selectors for IERC7891

Functional Signature	Keccak-256 Hash	Selector
mintParent(string)	f36d273d6e10fa164eeeff649e358305f719b1391fc64b390f59cf496008393f	0xf36d273d
mintSplit(uint256, uint8)	1ba5857f2babe6c049ec0842428d2e03ae24c815b9d420f92ed9fdbc6a1d5fab	0x1ba5857f
mintMerge(uint256, uint256[])	35d09c3045d626b2ccea4d95db4976eac2bb917a4c116b6757a41e1cfbd02978	0x35d09c30
sharePass(uint256, uint256, uint8)	dc45d3714c2624fee7a28399773097ecbce78e435786e584d2851bb48e81d901	0xdc45d371
burn(uint256)	42966c689b5afe9b9b3f8a7103b2a19980d59629bfd6a20a60972312ed41d836	0x42966c68

We then performed the XOR operation across all selectors to obtain the final interface ID:

$$0xf36d273d \oplus 0x1ba5857f \oplus 0x35d09c30 \oplus 0xdc45d371 \oplus 0x42966c68 = 0x43cb816b$$

Thus, the ERC-165 Interface ID we derived for IERC7891 is: 0x43cb816b.

2. Implementation in ERC7891.sol

To operationalize ERC-165 support, we implemented interface detection directly in our reference contract, ERC7891.sol. The supportsInterface function returns true when queried with the IERC7891 interface ID, enabling automated verification of ERC-7891 compliance. Additionally, we added a helper function, getInterfaceID(), to expose the calculated identifier for external validation and testing.

```
function supportsInterface(bytes4 interfaceId) public view
        override(ERC721Enumerable, IERC165) returns
        (bool) {
    return interfaceId == type(IERC7891).interfaceId
         || super.supportsInterface(interfaceId);
}

function getInterfaceID() external pure returns (bytes4) {
    return type(IERC7891).interfaceId;
}
```

Through this integration, ERC-7891 ensures full compatibility with Ethereum's standard detection framework.

V. Submission and Acceptance Process

The formal submission of our work follows the standardized EIP process, which serves not only as a procedural requirement but also as a mechanism for technical validation, community peer review, and formal recognition of the proposed protocol within the Ethereum ecosystem. The process comprises the following:

1. Drafting of ERC-7891

```
eip: 7891
title: Splitting and Merging of NFTs
description: Interface for hierarchical NFTs, enabling
splitting a single NFT and merging multiple NFTs
author: Nitin Bhagat (@nitin312), JongWook Bae, Su-Hyun
Lee
discussions-to:
https://ethereum-magicians.org/t/eip-7891-hierarchical-n
fts-with-splitting-and-merging/22986
status: Draft
type: Standards Track
category: ERC
created: 2025-02-15
requires: 721, 6150
```

We prepared the ERC-7891 proposal in full compliance with the Ethereum Improvement Proposal format defined in EIP-1[11]. The specification was written as erc-7891.md, incorporating all mandatory metadata fields as shown below and required structured sections including the abstract, motivation, specification,

rationale, backward compatibility, security considerations, and reference implementation.

2. Assets Folder Organization

To support transparency and reproducibility, we organized all implementation files according to the mandatory EIP directory structure.

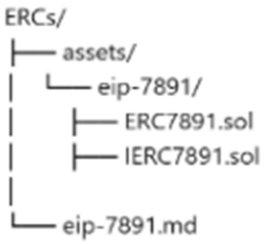

Fig. 2. Assets Folder Organization

As shown in <Fig. 2>, we created the folder ERCs/assets/eip-7891/, which contains the standard interface, IERC7891.sol and the complete reference implementation ERC7891.sol. This organization allows other researchers and developers to validate the correctness of the implementation and to compare it against existing NFT standards.

3. Pull Request Submission and Community Review

After finalizing the specification and reference implementation, we submitted ERC-7891 to the official Ethereum EIPs GitHub repository as Pull Request #930, titled “Add ERC: Splitting and Merging of NFTs”. The submission successfully passed all automated checks related to formatting, licensing, and directory validation, confirming its technical completeness.

In parallel, we introduced ERC-7891 on the Ethereum Magicians forum to invite feedback from the developer and research communities. The comments and discussions obtained through this process contributed to iterative refinement of the specification, improving its clarity, consistency, and interoperability. From an academic perspective, this open review process functioned as a peer-validation mechanism for protocol design.

4. Acceptance and Merging into EIP Repository

Following editorial evaluation and community discussion, ERC-7891 was officially accepted and merged into the Ethereum EIPs repository, as shown in <Fig. 3>. Its publication on the official Ethereum website confirms its inclusion in the standardization pipeline and current Draft status, progressing toward final approval[12].

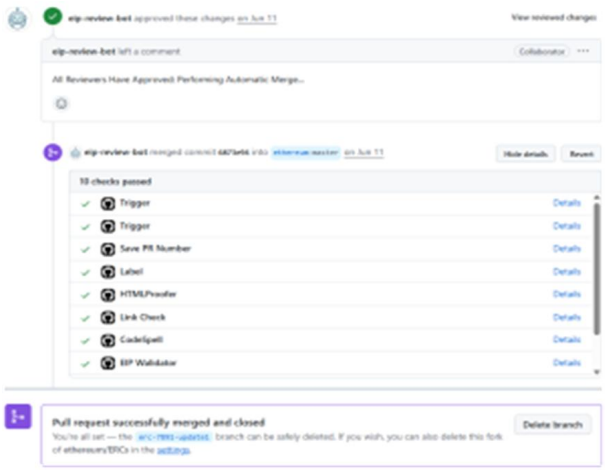

Fig. 3. Official Acceptance of ERC-7891 in the Ethereum EIPs Repository

This acceptance establishes ERC-7891 within the Ethereum standards ecosystem, enabling developers, decentralized applications, and marketplaces to adopt the protocol for hierarchical NFT restructuring.

VI. Conclusions

In this paper we introduced ERC-7891 Splitting and Merging of Hierarchical NFTs, a new Ethereum standard that addresses limitations in existing NFT protocols. Earlier standards such as ERC-721, ERC-1155 and ERC-6150 provided foundations for non-fungible ownership and hierarchical organization but lacked native mechanisms for dynamic restructuring and fractional ownership. These gaps restrict NFT applicability in domains where assets evolve, split or merge over time.

To address these gaps, we defined ERC-7891 as a

protocol-level extension that introduces standardized operations for splitting and merging within hierarchical NFT systems. The design introduces core functions mintParent, mintSplit, mintMerge, sharePass and burn, supported by events that ensure transparency and verifiable lineage tracking. The reference architecture, composed of the IERC7891 interface and the ERC7891.sol implementation, demonstrates integration with ERC-6150's hierarchical framework while maintaining ERC-721 compatibility, and includes ERC-165 interface detection for interoperability across decentralized applications. Furthermore, we documented the full EIP submission and acceptance process, confirming the technical validity and community relevance of ERC-7891 as an officially recognized Ethereum standard.

While ERC-7891 provides standardized mechanisms for splitting and merging within hierarchical NFTs the standard still depends on underlying protocols ERC-721, ERC-6150 and ERC-165 which increases implementation complexity and requires multi-layer compliance. Additionally this paper adopts a protocol-oriented analysis to illustrate multi-level hierarchical ownership and dynamic restructuring using representative real-world asset examples. As future work ERC-7891 can be extended and adopted as a native protocol for RWA DApps enabling more expressive modeling of asset structures in decentralized real estate systems, supply chain platforms and modular product systems. We also plan to conduct comprehensive quantitative and scenario-based experiments to evaluate the functional and structural improvements of ERC-7891 in comparison with ERC-404 and ERC-6150 based implementations. Furthermore, future work will investigate detailed threat models and response strategies for potential security risks, including privilege abuse, incorrect share propagation, and reentrancy attacks that may arise during critical protocol operations.

ACKNOWLEDGEMENT

This research was supported by Changwon National University in 2014.

REFERENCES

[1] B. Singh, K. Wongmahesak, and S. Chandra, "Transforming Digital Ownership and Assessing Role of Blockchain Technology and NFTs in Future Economy," in Practical Strategies and Case Studies for Online Marketing 6.0, IGI Global Scientific Publishing, pp. 343-368, 2025. DOI: 10.4018/979-8-3373-2058-8.ch015

[2] A. Semnani and G. Yang, "Non-Fungible Tokens (NFTs) Beyond Collectibles: A Comprehensive Review of Applications," SSRN Electronic Journal, December 2024. DOI: 10.2139/ssrn.5046792

[3] N. Bhagat, J. W. Bae and S. H. Lee, "A user friendly NFT platform for Digital Assets," Proceedings of the Korean Society of Computer Information Conference, pp. 447-450, July 2023.

[4] Keegan Lee, msfew, et al., "EIP-6150: Hierarchical NFTs," https://eips.ethereum.org/EIPS/eip-6150, accessed August 2025.

[5] N. Bhagat, J. W. Bae, and S. H. Lee, "A Blockchain Approach to Product History Tracking Using Hierarchical NFTs," Journal of the Korea Society of Computer and Information, Vol. 30, No. 9, pp. 87-97, September 2025. DOI: 10.9708/jksci.2025.30.09.087

[6] W. Choi, J. Woo, and J. W. K. Hong, "Fractional non-fungible tokens: Overview, evaluation, marketplaces, and challenges," International Journal of Network Management, Vol. 34, No. 4, e2260, June 2024. DOI: 10.1002/nem.2260

[7] H.-W. Long and Y.-W. Si, "Token Fungibility Duality: Technical and Graphical Analysis on 404 Standards," 2024 IEEE International Conference on Blockchain (Blockchain), pp. 252-259, August 2024. DOI: 10.1109/Blockchain62396.2024.00040

[8] J. W. Bae, N. Bhagat, and S. H. Lee, "Hierarchical NFT using Parent-Child Structure," Journal of The Korea Society of Computer and Information, vol. 29, no. 2, pp. 127-136, February 2024. DOI: 10.9708/jksci.2024.29.02.127

[9] N. Bhagat, J. W. Bae, and S. H. Lee, "Dynamic Management of Hierarchical NFTs: Efficient Splitting and Merging," Journal of the Korea Society of Computer and Information, vol. 30, no. 2, pp. 73-82, February 2025. DOI: 10.9708/jksci.2025.30.02.073

[10] C. Reitwießner, N. Johnson, et al., "EIP-165: Standard Interface Detection," https://eips.ethereum.org/EIPS/eip-165, accessed November 2025.

[11] M. Becze, H. Jameson, et al., "EIP-1: EIP Purpose and Guidelines," https://eip.directory/eips/eip-1, accessed August 2025.

[12] N. Bhagat, J. Bae, and S. H. Lee, "ERC-7891: Splitting and

Merging of NFTs," https://eips.ethereum.org/EIPS/ eip-7891, accessed November 2025.

Authors

Nitin Bhagat received the B.E and M.E degrees in Computer Engineering from Pokhara University, Nepal, in 2004 and 2017, respectively. He joined the faculty of Computer Engineering as a Lecturer at Purbanchal University, Biratnagar, Nepal, in 2007. He is currently a Research scholar in the blockchain technology at Changwon National University in Korea. He is interested in blockchain technology, NFT web platform design and evaluation.

JongWook Bae received the B.S. in Computer Science from Changwon University, Korea in 2001, followed by his Master's and Doctoral degrees in the same department in 2005 and 2012, respectively. He currently is a lecturer in the Department of Computer Science at Changwon National University. He is interested in image processing and blockchain.

Su-Hyun Lee received the B.S. in Computer Science from Kwangwoon University, Korea in 1987. He received the M.S. and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology(KAIST), Korea, in 1989, 1994, respectively. Dr. Lee is a Professor in the Department of Computer Engineering, Changwon National University since 1996. He is interested in computer algorithm, programming languages, compiler, and blockchain.