JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 30 No. 9, pp. 87-97, September 2025
https://doi.org/10.9708/jksci.2025.30.09.087

A Blockchain Approach to Product History Tracking Using Hierarchical NFTs

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 a blockchain-based product tracking framework using hierarchical Non-Fungible Tokens(NFTs), enabling modular representation and traceability. In contrast to existing flat NFT structures, our system introduces a modular and traceable hierarchical model for complex assembly products. A key innovation of the system is a two-phase transfer mechanism where first, structural reassignment updates the NFT hierarchy to reflect new relationships and then, recursive ownership transfer ensures all subordinate NFTs are transferred in a depth-first manner. This design ensures part-level traceability, transparency, and accountability throughout a product's lifecycle. The system offers practical benefits for domains such as automotive, electronics, and healthcare, where managing modular assets and detailed traceability is critical. By extending the NFT framework to support structural reallocation and recursive transfers, the proposed model enhances product verification, maintenance tracking, and ownership transparency.

▸ **Key words**: Hierarchical NFTs, Blockchain, MetaMask, Smart Contracts, Product Traceability

[요 약]

본 논문은 계층적 대체불가 토큰(NFT)을 활용한 블록체인 기반 제품 추적 프레임워크를 제시한다. 복합 조립 제품을 모듈 단위로 표현하고 추적 가능성을 확보함으로써, 기존의 평면적 NFT가 지닌 한계를 극복한다. 본 시스템의 주요 아이디어는 NFT 계층에 구조적이고 소유권 일관성을 보장하는 2단계 전송 메커니즘에 있다. 첫 단계에서 구조적 재할당을 통해 새로운 관계를 반영하도록 NFT 계층을 갱신한다. 다음 단계에서 재귀적 소유권 이전을 통해 모든 하위 NFT를 깊이 우선 방식으로 전송한다. 이러한 설계는 제품 수명 주기 전반에 걸쳐 부품 수준의 추적성, 투명성 및 책임성을 보장한다. 이 시스템은 모듈식 자산 관리 및 세부적인 추적성이 중요한 자동차, 전자, 의료와 같은 분야에 이점을 제공한다. 제안된 모델은 구조적 재할당 및 재귀적 전송을 지원하도록 NFT 프레임워크를 확장함으로써 제품 검증, 유지 관리 추적 및 소유권 투명성을 향상시킨다.

▸ **주제어**: 계층적 NFT, 블록체인, 메타마스크, 스마트계약, 상품 추적

• 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. 07. 14, Revised: 2025. 09. 07, Accepted: 2025. 09. 10.

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

I. Introduction

In today's global supply chains, ensuring product authenticity and traceability is critical[1]. Traditional centralized tracking systems lack transparency, scalability, and are prone to tampering. These limitations are especially problematic for complex products, where tracking individual components and their integration into final assemblies becomes increasingly challenging.

Blockchain technology provides a decentralized, tamper-proof ledger ideal for product lifecycle tracking[2]. Non-Fungible Tokens(NFTs), as unique digital assets on the blockchain, can act as digital twins of physical products, embedding metadata, ownership, and maintenance records[3]. This ensures auditable product histories, enhancing quality control and consumer trust. However, conventional NFTs lack the capability to express hierarchical relationships between components of complex products.

To address this gap, this paper proposes a blockchain-based product history tracking using hierarchical NFTs. Each NFT represents a product or component, with metadata stored on interplanetary file system(IPFS). Final products are represented as parent NFTs, while individual components are modeled as child NFTs. The model supports modular tracking and ownership transfer across all layers of a product's lifecycle. The combination of hierarchical modeling and smart contract logic enables product registration, part-level tracking, secure transfers, and tamper-proof historical records.

This paper presents a blockchain-based methodology for representing complex products and tracking their history using hierarchical NFTs. It further explores the potential benefits, challenges, and implications of this concept in various areas. Chapter 2 provides the theoretical background, explains the existing blockchain based history tracking, while Chapters 3 and 4 detail the design of the proposed scheme and implementation. Chapter 5 provides conclusion and summary of our contributions.

II. Theoretical Background

1. Product History Tracking with Blockchain

In modern supply chains, the ability to track a product's entire lifecycle, from manufacturing to end-user delivery, is essential for ensuring transparency, authenticity, and accountability[4]. Traditional tracking systems often suffer from fragmented and inconsistent standards, making it difficult to verify product authenticity and component origins. This challenge is particularly critical in sectors like pharmaceuticals, electronics, and automotive manufacturing, where precision and trust are paramount.

Non-Fungible Tokens, with their immutable and verifiable nature on the blockchain, offer a transformative solution for product traceability. Acting as digital twins of physical items, NFTs can capture essential data such as manufacturer details, production dates, ownership records, and component specifications[5]. Key lifecycle events like ownership transfers, part replacements, or inspections can be securely recorded on-chain. This decentralized and tamper-proof record keeping enhances transparency, reduces counterfeiting risks, and supports regulatory compliance through comprehensive auditability.

2. Hierarchical NFTs

Hierarchical Non-Fungible Tokens extend the ownership and uniqueness features of traditional NFTs by introducing a structured parent-child relationship[6]. Unlike flat NFTs, which represent singular items, this model allows token to be linked in multi-level structures, mirroring real-world product compositions.

In the context of product history tracking, this parent-child structure is particularly valuable. In the automotive sector, it enables component-level

tracking where engines, batteries, or tires are registered as child NFTs, preserving their replacement or repair history. In electronics manufacturing, hierarchical NFTs can represent modular devices, such as smartphones, where displays, processors, and sensors are tracked individually.

By capturing the relationships between interconnected assets, hierarchical NFTs support detailed lifecycle tracking, decentralized data storage, and flexible ownership structures. This makes well-suited for applications requiring multi-layered asset representation for product history management.

3. Existing Blockchain Based History Tracking

Dietrich et al. introduced a blockchain-based traceability architecture using a blueprint-based token model, where tokens of the same type are grouped and minting conditions define object assemblies. Their system enables granular tracking of object-related events within supply chains. However, while effective in mapping discrete object flows, it lacks support for hierarchical NFT structuring and ownership modeling essential for modular product assemblies[7].

Gebreab et al. proposed an NFT-based traceability and ownership management system for medical devices using blockchain technology, where NFTs represent device attributes, certifications, and ownership. The system utilizes Ethereum smart contracts and IPFS to ensure transparency and authenticity. While this model effectively captures individual devices and their histories, it lacks the capability to represent modular, multi-level product assemblies[8].

Hsing-Chung Chen et al. proposed a trust-chain framework for integrated circuit(IC) traceability using a Hierarchical Content Identifier Mechanism(HCIDM). Their system links ICs, modules, and systems through Merkle Directed Acyclic Graphs(DAGs), enabling data-level traceability and integrity verification. However, while structurally similar to parent-child relationships, their model focuses on static content linkage rather than supporting dynamic reallocation within modular product hierarchies[9].

In contrast, our proposed blockchain-based product tracking scheme uses hierarchical NFTs to represent product and its individual components. Each product is minted as NFT, along with metadata details and images stored on IPFS. Smart contracts enforce parent-child relationships, supporting multi-level hierarchies. Our modal-based interface allows the transfers of NFTs between registered companies and users, with all ownership changes recorded on-chain. This system ensures transparency by allowing users to trace components, verify ownership history, and understand product structure.

III. Design of the Proposed Scheme

1. Structure of the Proposed Hierarchical NFT

Our proposed scheme is built upon the Ethereum token standard ERC-6150, which introduces a hierarchical framework for modeling relationships among NFTs[10]. Each final product is minted as a parent NFT, while its partial products are registered as child NFTs, forming a tree-like structure.

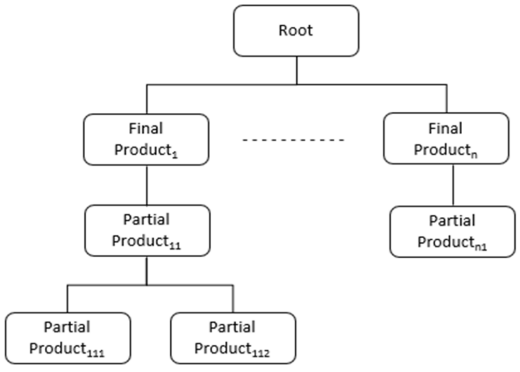

Fig. 1. Proposed Hierarchical Scheme

As illustrated in <Fig. 1>, the hierarchy begins from a universal root node, linking to top-level

NFTs representing final products. Each final product NFT can then branch into one or more partial product NFTs, which themselves may further branch into deeper levels, forming the parent-child structure.

To enhance traceability and ensure data integrity, each NFT's metadata is securely stored on the IPFS. This metadata includes critical fields such as product name, serial number, category, date, and associated images, forming a tamper-proof and decentralized repository of product information. By extending the ERC-721 token standard, ERC-6150 enables relationships in which each NFT is aware of its parent and children, allowing for dynamic and intuitive relationship management[11].

This hierarchical structure serves as the operational backbone for the subsequent stages of our proposed system, encompassing final product minting, partial product minting, hierarchical transfers, ownership transfer, historical tracking, and token burning, thereby enabling reliable and transparent product history tracking on the blockchain.

To ensure compatibility with existing NFT ecosystems, ERC-6150 extends the widely adopted ERC-721 standard, preserving core functionalities such as ownership and metadata referencing. Furthermore, ERC-6150 supports ERC-165 interface detection, enabling integration and interoperability with ERC-721-based platforms and other NFT ecosystems.

2. Product Registration Scheme

The hierarchical relationship between NFTs we propose is established at the time of minting by specifying the parent token ID. If the parent token ID is 0, the token is recognized as a final product and is anchored to Root node. Otherwise, it is considered a partial product, forming a child NFT under the specified parent. This setup forms a tree-like structure where each token maintains bidirectional links to both its parent and children.

2.1 Dual Link

To maintain a consistent hierarchy navigation, we introduce a dual link mechanism during the NFT minting process. This structure ensures that every parent NFT tracks its associated components using Children[Tid] array, which maintains an indexed array of all child NFTs linked to it. Simultaneously, each child NFT references its immediate parent through Parent array. This two-way linkage is uniformly applied to both final product and partial product minting processes.

2.2 Final Product NFT Minting

In our proposed scheme, a final product is defined as an NFT minted with the root node as its parent. Each node in the system is uniquely identified by a token ID, Tid. As shown in <Fig. 2>, when the parent token ID is set to 0 during minting, the token Tid_1 is treated as a final product and anchored to the root node, Tid_0.

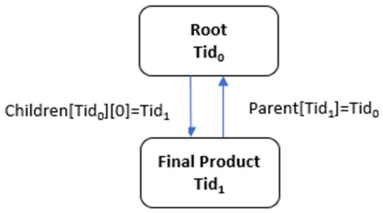

Fig. 2. Final Product NFT minting

This allows to establish a dual link, where the root node tracks the final product as its child using Children[Tid_0][0] = Tid_1, and the final product tracks its parent as Parent[Tid_1] = Tid_0. This bidirectional linkage establishes Tid_1 as the starting point of a product assembly and allows it to serve as a parent for additional components.

2.3 Partial Product NFT Minting

We define a partial product as a token minted with a valid parent token ID. The minting of partial product NFTs extends the hierarchical structure by linking newly created tokens to pre-existing ones. A dual link mechanism allows the parent NFT to link its newly minted child in the Children[Tid]

array, while the child NFT references its parent via Parent[Tid].

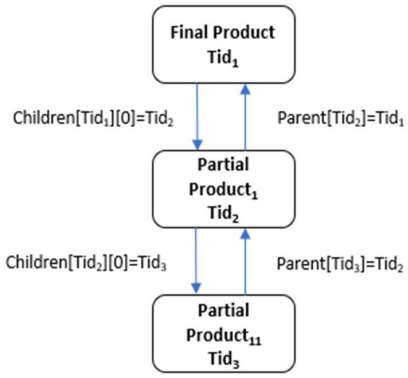

Fig. 3. Partial Product NFT Minting

As shown in <Fig. 3> Tid_2 is created as a child of Tid_1, and Tid_3 as a child of Tid_2, forming a multi-level hierarchy. The minting process supports hierarchical expansion and allows each token to serve as a parent to future tokens. This structure preserves parent-child relationships, and supports traceability within nested product hierarchies.

3. Product Transfer Scheme

Transferring a product NFT along with its components is essential for maintaining ownership accuracy and structural integrity within the hierarchy. In our proposed scheme, child NFTs are initially assigned to their respective parent NFTs. During a product transfer, all associated child NFTs are relocated along with the parent. This is achieved through two key processes: structural reassignment and recursive ownership transfer. This ensures not only the accurate repositioning of NFTs within the hierarchy but also the secure handover of ownership across all levels of associated components.

3.1 Structural Reassignment

The transfer begins with the structural reorganization of the NFT hierarchy, specifically by changing the parent of the token to be transferred. This process first updates the parent relationship of the NFT identified by tid, linking it to a new parent token targetId, which belongs to the recipient. This redefines the token's position in the product tree by setting Parent[tid] = targetId, thereby linking the transferred token as a child under the recipient's token.

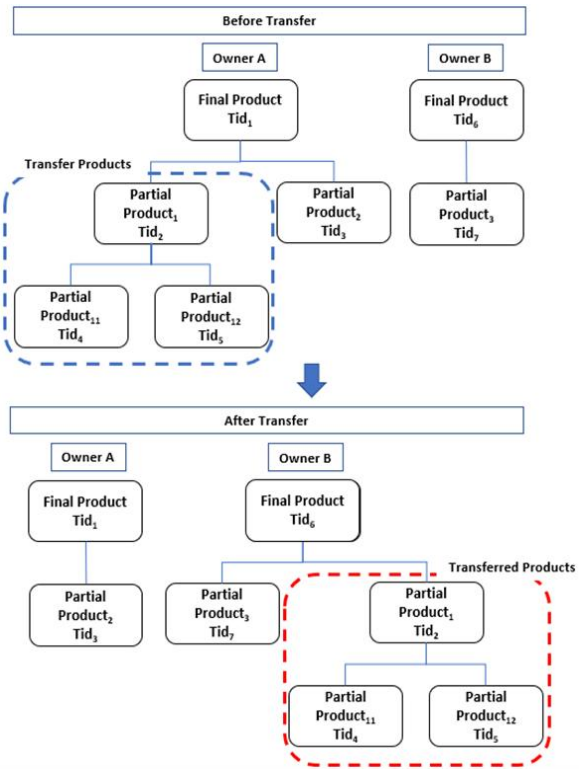

Fig. 4. Hierarchical Product Reallocation Before and After Transfer

Next, the child array of the new parent is updated so that child[targetId] includes tid. This reassignment embeds the transferred token into the recipient's existing hierarchy, structurally integrating the product or part into the new owner's structure. The <Fig. 4> shows the ownership hierarchy before and after transferring a product NFT and its components.

3.2 Recursive Ownership Transfer

Once the structural reassignment is complete, the actual transfer of ownership takes place. If the token tid has no children, the transfer is performed directly by transferring the token from the current owner to the owner of targetId. However, if tid

represents a token with one or more child NFTs, then the recursive transfer is needed to maintain the integrity of hierarchical structure.

In recursive transfer, each descendant NFT is first transferred to the new owner through a depth-first traversal[12]. Only after all children are transferred, the parent token, tid itself is transferred. Each transfer is executed using transferFrom(tid, targetId), moving ownership from the current owner to the owner of targetId.

4. Burn Scheme

When deleting NFTs, a product NFT and all of its associated sub-components are deleted. The deletion process begins at the selected token ID (tid) and traverses downward through its children. Each child NFT undergoes recursive transfer evaluation. If it has its own children, the process continues until leaf nodes are reached. Only tokens identified as leaves are eligible for immediate burning. This ensures that all descendant tokens are safely removed before their parent is deleted.

IV. Implementation of the Proposed Scheme

1. A Smart Contract of the Proposed System

Our "ProductHistory" smart contract serves as the backbone for the proposed system for hierarchical NFT-based product tracking. It inherits the ERC-721 Enumerable and ERC-6150 protocols to enable ownership tracking and establish parent-child relationships among tokens. In our model, user can mint a final product as a parent NFT, while its associated components are allowed to be registered as child NFTs linked to their respective parents.

Our hierarchical approach tracks the product by maintaining parent-child relationships, and also manages the history of all transactions occurred within the system. The contract facilitates operations such as product registration, user registration, product transfer, and product deletions while maintaining the traceability of products and integrity of the hierarchy. This system ensures users maintaining detailed record of each product and its associated parts.

Our system secures metadata storage by integrating IPFS with Ethereum smart contracts. IPFS provides content-addressable storage, and the immutability of hashes guarantees data integrity. Smart contracts enforce strict permissions for NFT operations, ensuring that only authorized users can interact with the metadata.

2. Product Registration

2.1 Final Product Registration

As illustrated in <Fig. 5>, our system enables the registration of a final product through a user-friendly interface. Users begin by filling out essential product information, including Name, Serial Number, Category, Date, and Description and Choose File, and log into their MetaMask wallet. Upon clicking the [Minting] button, an exception check is made to ensure that all the necessary fields are filled and triggers the smart contract function 'mintWithParent()' with parentId = 0. This token is a top-level NFT with no parent.

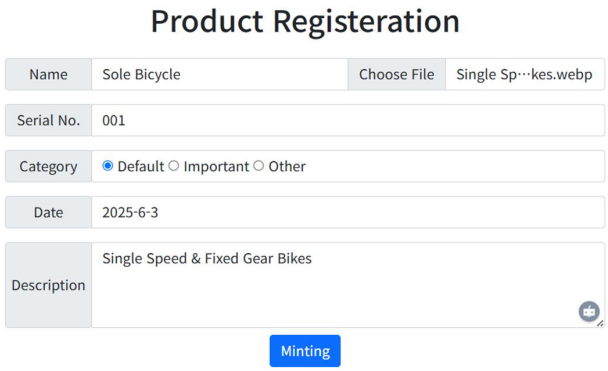

Fig. 5. Final Product NFT Minting

Internally, the smart contract logic uses '_safeMintWithParent()' function inherited from the ERC-6150 standard to mint the NFT and assign it to the blockchain. This ensures that the final product is anchored to the root of the hierarchy.

```
function mintWithParent(uint256 parentId, string memory
                        _tokenURI) public payable {
    tokenIds.increment();
    uint256 tokenId = tokenIds.current();
    _safeMintWithParent(msg.sender, parentId, tokenId);
    tokenURIs[tokenId] = _tokenURI;

    approve(address(this), tokenId);
    if (parentId > 0) {
        address parent = ownerOf(parentId);
        if (msg.sender != parent){
            ERC721(address(this)).safeTransferFrom(msg.sender,
            parent, tokenId);
        }
    }
}
```

2.2 Product List View

Once products are registered, they appear in the Product List, as illustrated in <Fig. 6>. This page lists the products owned by specific user and allows them to monitor and control their products. Users can register sub-components by clicking Part Reg.

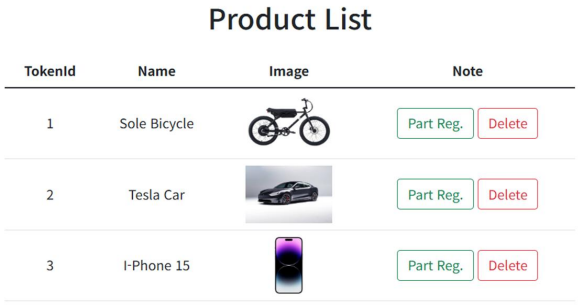

Fig. 6. List View of Final Product Minted NFTs

2.3 Partial Product Registration

The Part Reg. button opens a modal window as shown in <Fig. 7> where users can register partial product by entering part specific metadata such as Name, Serial Number, Date, Category, and Description along with a file upload. When clicking on [Minting] button, the same 'mintWithParent()' function is invoked within smart contract with parentId > 0. Here the contract not only mints the partial product but also establishes a parent-child link between the product NFTs.

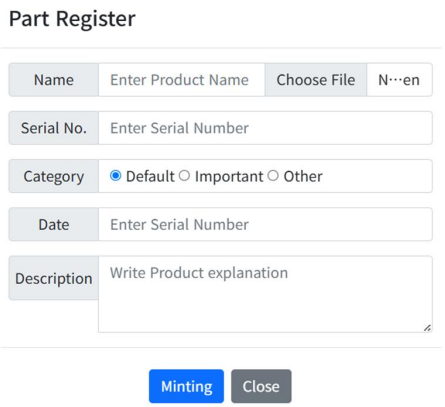

Fig. 7. Partial Product NFT Minting

3. Product Details and History

Once a user selects a product from the "Product List" interface, they are redirected to the "Detail Contents" page, which displays an overview of the selected tokenized product and its metadata.

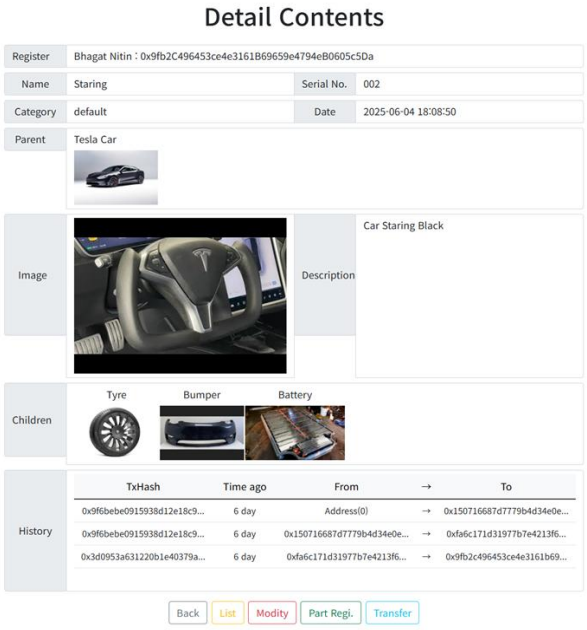

Fig. 8. Product Overview with Parent-Child Relationships

As illustrated in <Fig. 8>, the page displays key attributes such as the product name, serial number, category, registration date, description, and an image stored on IPFS. This detail page also shows parent child relationship of a product within the hierarchy. The image item shows a detailed image of the viewed part, with the image of the parent part at the top and child part appears at the bottom. The selected NFT shown in figure is a partial product, so the Parent field shows its corresponding final product “Tesla Car”, while the Children field lists its associated partial products “Tyre,” “Bumper,” and “Battery”. Clicking on each image takes you to the detail page for that part.

The History section at the bottom of the Detail Contents page offers user a clear timeline of NFT ownership changes. The first entry records the minting event, assigning ownership from Address(0), the system origin to 0x15071668d7..., marking the NFT’s creation. Next, the NFT is transferred to the parent NFT’s owner (0xfa6c171d31...), establishing its place in the product hierarchy. The third entry shows a transfer from 0xfa6c171d31... to 0x9fb2C496..., indicating a new owner. Each transaction is securely recorded on-chain with a unique transaction hash, TxHash, ensuring immutable proof of ownership and product lineage.

```
async function getLogs(tid, topic) {
    const logs = await web3.eth.getPastLogs({
        fromBlock: '0x0',
        toBlock: 'latest',
        address: CONTRACTADDRESS,
        topics: [web3.utils.sha3(topic)],
    });
    return logs.filter(log => tid == parseInt(log.topics[3], 16));
}
```

Logs related to NFT minting and transfers are permanently recorded on the Ethereum blockchain, enabling historical traceability. getPastLogs() method queries blockchain event data from the contract address across all blocks. The retrieved logs are then filtered using logs.filter(), which isolates only those entries corresponding to the selected token ID. This filtering process extracts only relevant ownership changes and displays a transaction history for each NFT from its creation to the current owner.

4. Product Transfer

When a user initiates the transfer of a product by selecting a token from the “Detail Contents” page and clicking the [Transfer] button, a modal window “Select Transfer TokenID” appears as shown in <Fig. 9>. This shows a hierarchical view of all products and their associated parts registered under the selected recipient. The process involves identifying the target location within the recipient’s product hierarchy where the selected part should be integrated.

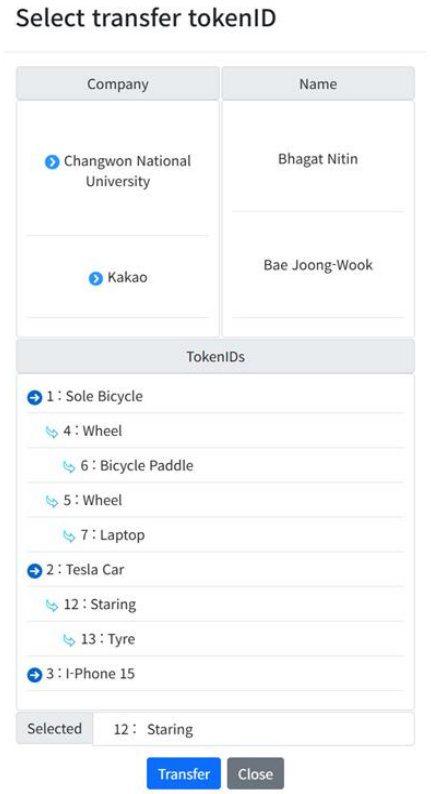

Fig. 9. Product Selection and Transfer

The user first selects a recipient company and then chooses a specific user under that company. Upon selection, all root NFTs associated with the

selected user are displayed hierarchically, showing both parent and child tokens. The user selects the appropriate destination token ID (targetId) under which the product will be relocated. After confirming the selection and clicking on [Transfer] button, the system executes 'hierarchyTransfer()' function within our smart contract, which performs the transfer through two key functions.

```
function hierarchyTransfer(uint256 tid, uint256 _targetId)
                            public payable {
    structuralReallocate(tid, _targetId );
    recursiveTransfer(tid, _targetId );
}
```

First, 'structuralReallocate()' performs structural reassignment that updates the token's parent reference to reflect its new position in the hierarchy and modifies child arrays of both the old and new parent tokens accordingly. Specifically, the token is removed from the child list of its previous parent and added to the child list of the newly assigned parent. This reallocation ensures each component remains in correct position within the recipient's hierarchy after transfer.

```
function structuralReallocate(uint256 tokenId, uint256
                                newParent) public payable{
    requireMinted(tokenId);
    uint256 oldParent = parentOf[tokenId];
    childrenOf[oldParent].pop();
    parentOf[tokenId] = newParent;
    childrenOf[newParent].push(tokenId);
    indexInChildrenArray[tokenId]=childrenOf[newParent].length;
}
```

Next, 'recursiveTransfer()' transfers ownership of all descendant NFTs before transferring the parent token itself.

```
function recursiveTransfer(uint256 tid, uint256 _targetId)
                            public payable {
    uint256 [] memory children = childrenOf(tid);
    for (uint256 i = 0 ; i< children.length ; i++) {
        recursiveTransfer(children[i], _targetId );
    }
    transferFrom(tid, _targetId );
}
```

Here, the function first retrieves all child tokens of the given NFT and recursively calls itself for each child. Only after all children are transferred does it transfer the parent token. This ensures that the hierarchy is preserved and all components are reassigned in the correct order.

5. Deletion of Product

When the user clicks the [Delete] button in <Fig. 6>, 'recursiveBurn()' function is called to initiate the deletion process. This function recursively traverses the NFT hierarchy, starting from the selected token ID, and deletes all associated child tokens before removing the parent token itself. By ensuring that no part of the product tree remains undeleted, this approach maintains data integrity and prevents orphaned records in the system.

```
function recursiveBurn(uint256 tid) public {
    uint256 [] memory children = childrenOf(tid);
    for (uint256 i = 0 ; i< children.length ; i++) {
        recursiveBurn(children[i]);
    }
    if (isLeaf(tid)) _safeBurn(tid);
}
```

6. Empirical Evaluation

We conducted empirical evaluation by deploying the smart contract on the local Ethereum blockchain simulated using Ganache. The experiments covered the core functions for parent NFT minting, child NFT minting, transfer NFT, and burn NFT. Each operation was executed multiple times to obtain consistent average measurements. We collected two key metrics, gas used for operation and average time recorded between transaction submission and confirmation.

Table 1. Average Gas and Time

Operation	Gas Used	Time(ms)
Mint Parent NFT	298,258	30
Mint Child NFT	317,937	36
Transfer NFT	158,343	26
Burn NFT	70,724	29

The results, summarized in Table 1, show that minting operations consume more gas than transfers or burns, but all remain within practical limits. Confirmation times averaged only a few milliseconds, showing that the system can achieve responsive performance under simulated blockchain conditions.

To further evaluate scalability, we tested NFT transfer across hierarchies by increasing its depth. As summarized in Table 2, gas consumption rises proportionally with the number of nested NFTs, while confirmation time increases moderately. This indicates that even for complex product assemblies with multi-level hierarchies, the framework remains computationally manageable and efficient.

Table 2. Scalability with Hierarchical Size

Hierarchy Size	Avg. Gas Used	Avg. Time(ms)
Level 1(1 NFT)	158,343	26
Level 2(3 NFTs)	320,737	28
Level 3(7 NFTs)	439,560	38
Level 4(15 NFTs)	802,122	169

The gas used for minting and transfer for a flat ERC-721 NFT is approximately 68,355 and 55,324 respectively[13]. In contrast, our system shows higher gas usage because ERC-6150 extends ERC-721 by incorporating additional hierarchical functionalities. Each minting and transfer operation in our framework not only records ownership but also updates parent-child mappings and maintains structural consistency across the hierarchy. These added computations and storage writes increase gas consumption compared to flat NFTs. Even though the gas consumption is higher, the execution time is reasonably fast. Moreover, as the depth increases, confirmation time grows only slightly, showing that the system can still handle complex, multi-level product assemblies efficiently.

V. Conclusions

In this paper, we proposed a blockchain-based product history tracking system using hierarchical NFTs compliant with the ERC-6150 standard. Our approach enables modeling of complex products as parent-child NFT relationships, where finished products are represented as parent NFTs and their components as child NFTs. This structure allows for clear representation, tracking, and reallocation of each component within a product assembly.

While existing blockchain-based history tracking systems typically rely on flat, event-driven architectures to trace asset movements, they often fall short in capturing the structural complexity and inter-dependencies of real-world products. In contrast our proposed system addresses this gap by introducing this hierarchical NFT model that supports parent-child relationships between final products and their sub-components.

We proposed a two-phase transfer mechanism that ensures structural and ownership consistency across NFT hierarchies. It combines structural reassignment with recursive ownership transfer to maintain integrity during product handovers. Additionally, our system maintains detailed product history for each component, preserving lifecycle records across the supply chain throughout the hierarchy which lacks in the existing tracking systems.

Our hierarchical NFT-based product history tracking system can benefit for industries such as automotive, electronics, and healthcare sectors where modular assemblies require secure, component-level traceability and tamper-proof history.

While the proposed framework models hierarchical NFT using DFS for recursive operations for transfer and burn, we acknowledge that this approach may lead to increased gas costs in deeply nested hierarchies. As future work, we aim to address this scalability limitation and enhance the system by developing visual tools for

hierarchy navigation and standardized auditing features to further promote the adoption of blockchain-based traceability systems.

ACKNOWLEDGEMENT

This research was supported by Changwon National University in 2025~2026.

REFERENCES

[1] A. Sobowale, R. Okon, G. Nzeako et al., "Ensuring product authenticity and traceability with blockchain in supply chains," World Journal of Advanced Research and Reviews, Vol. 24, No. 2, pp. 1017-1038, November 2024. DOI: 10.30574/wjarr.2024.24.2.3413

[2] P. Azevedo, J. Gomes, and M. Romão, "Supply chain traceability using blockchain," Operations Management Research, Vol. 16, pp. 1359-1381, April 2023. DOI: 10.1007/s12063-023-00359-y

[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, Vol. 31, No. 2, pp. 447-450, July 2023.

[4] M. Budler, B. F. Quiroga, and P. Trkman, "A review of supply chain transparency research: Antecedents, technologies, types, and outcomes," Journal of Business Logistics, Vol. 45, No. 1, pp. 1-28, December 2023. DOI: 10.1111/jbl.12368

[5] P. Kuruppuarachchi and A. McGibney, "Non-Fungible Token (NFT) Platform for Digital Twin Trust Management and Data Acquisition," Springer, Vol. 2022, pp. 264-277, February 2024. DOI: 10.1007/978-3-031-51643-6_19

[6] 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

[7] F. Dietrich, L. Louw, and D. Palm, "Blockchain-Based Traceability Architecture for Mapping Object-Related Supply Chain Events," Sensors, Vol. 23, No. 3, pp. 1410, January 2023. DOI: 10.3390/s23031410

[8] S. A. Gebreab, H. R. Hasan, K. Salah, and R. Jayaraman, "NFT-Based Traceability and Ownership Management of Medical Devices," IEEE Access, Vol. 10, pp. 126394-126411, December 2022. DOI: 10.1109/ACCESS.2022.3226128

[9] H. C. Chen, B. Irawan, P. Y. Hsu et al., "An Implementation of Trust Chain Framework with Hierarchical Content Identifier Mechanism by Using Blockchain Technology," Sensors, Vol. 22, No. 13, pp. 4831, June 2022. DOI: 10.3390/s22134831

[10] K. Lee, Msfew, Kartin, and Qizhou, "ERC-6150 Hierarchical NFTs," *https://eips.ethereum.org/EIPS/eip-6150* accessed 2025.08.19

[11] 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

[12] G. Rathi and S. Goel, "Applications of Depth First Search: A Survey," International Journal of Engineering Research & Technology (IJERT), Vol. 2, No. 7, pp. 134, July 2013. DOI: 10.17577/IJERTV2IS70557

[13] W. Choi, J. Woo, and J. W. K. Hong, "Gas cost analysis of fractional NFT on the Ethereum blockchain," IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 1-6, July 2023. DOI: 10.1109/ICBC56567.2023.10174920

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.