Research Paper **Open Access**

Mohammad Darabseh* and João Poças Martins

Protecting the intellectual property of built environment designs using blockchain technology

DOI 10.2478/otmcj-2023-0011
Received:April 27, 2022; accepted: July 16, 2023

Abstract: Digital design and modelling for construction assets was an important step towards improving the construction process overall by improving its efficiency and productivity, reducing the time needed for design editing and rework. However, digital forms of designs are prone to cyber threats and misuse by unauthorised authors. This is a common problem in different industries wherein digital asset management is inefficient due to its centralisation and depends on humans following a certain procedure. Blockchain is an emerging technology capable of transferring digitally produced information into a transferable digital value by storing the information or an identifying signature for the information on a blockchain ledger. Information on the blockchain ledger is immutable and stored in a decentralised system, making it a permanent record. This article investigates the possibility of protecting the intellectual property of built environment designs using blockchain technology. The article presents a solution for generating a double-fingerprint identity for the Industry Foundation Classes (IFC), a common format for built environment design exchange, followed by deployment of a smart contract on the Ethereum public blockchain to store the design fingerprint along with design meta-data, such as information about the the owner and version, as a non-fungible token, a unique asset format stored on a blockchain ledger.

Keywords: construction, Building Information Modelling (BIM), IFC, blockchain, intellectual property, ownership, smart contract

***Corresponding author: Mohammad Darabseh,** CONSTRUCT – GEQUALTEC, Faculty of Engineering (FEUP), University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal, E-mail: darabseh@outlook.com
João Poças Martins, CONSTRUCT – GEQUALTEC, Faculty of Engineering (FEUP), University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal

1 Introduction

Building information modelling (BIM) has positively affected the construction development process by creating higher-quality accurately built assets (Chan et al. 2019). However, Building Information Modelling (BIM) comes short in the context of data use control and governance. BIM collaborative approach is an open content-sharing environment, in which it is hard to track data ownership (Beach et al. 2017). Blockchain provides a standard route to creating digital assets by generating non-fungible tokens (NFTs) (Kugler 2021) for digital collectables to serve as authenticity and ownership certificates.

In construction projects, collaborative design is the norm nowadays; it helps stakeholders minimise risks such as reworks. However, collaborative work environments face several challenges. Legal challenges include intellectual property rights management, model ownership and copyright violations (Alreshidi et al. 2017). This research investigates how blockchain can help control the construction design process to overcome the challenges of traditional BIM governance.

In this article, a design model in the Industry Foundation Classes (IFC) format was considered a collectable digital asset, and the process of issuing ownership certificates for designs was investigated.

The article consists of four parts. The state-of-the-art section describes the problem and presents the technology on which a solution is built. This section also reviews examples of similar solutions from the literature. The second part unravels the component of the fingerprint designed to capture BIM designs. The third part shows the NFT smart contract development process, in addition to the tools used. The fourth part validates the designed fingerprint.

2 State of the art

This section provides a brief synopsis of the topics integrated into this article, covering the need for the solution

presented. The technology used to develop this solution is described. Examples of similar solutions from the literature are presented.

Although no previous studies have assessed the use of blockchain for intellectual property protection in the built environment, blockchain has been used for this purpose in other activities, as described in Section 2.4 on 'Blockchain for intellectual property protection.'

2.1 Built environment designs and intellectual property infringement

The use of digital design and authoring tools instead of the manual design approach improved productivity and reduced the complexities in the design development process. Indeed, collaborative construction design and BIM-based design approaches improved the construction design process and construction projects overall by solving design clashes before facing them in the construction stage and by reducing the time required for design rework tasks (Adibfar et al. 2020). However, digital design files are prone to cyber threats, resulting in sensitive data leaks or design copyright infringement, especially since design files developed using BIM are data-rich models containing reusable elements (Sardroud et al. 2018). An infringement happens when a construction design is used wilfully or unintentionally without authorisation from the legal owner. However, determining ownership in a collaborative environment, wherein digital assets result from the efforts undertaken by multiple stakeholders, is difficult. A simplified ownership concept within the collaborative work can have three ownership levels: (1) contribution, which refers to a limited level of involvement in the whole development; (2) authorship refers to a substantial contribution to the development of the digital built environment asset; the authors are legally liable for the content they produce; however, they do not own the rights to control the whole asset; (3) ownership refers to the ultimate right for controlling a built environment digital asset, including transferring ownership or delegating authorship to perform modifications on the digital asset (Darabseh and Martins 2021).

2.2 Overview of blockchain technology

Blockchain is a combination of decentralised data storage technology and cryptography. The implementation of this technology means introducing a decentralised and distributed ledger connected using cryptographic hashes. The data stored in a blockchain ledger is stored in block forms and chained together cryptographically, hence the name blockchain (Crosby et al. 2016). Data stored in the blockchain ledger is immutable, which protects it against change, thus providing an indisputable digital source of information that can be utilised in several areas, such as healthcare (Kuo et al. 2017), government (Ølnes et al. 2017) and finance (Tapscott and Tapscott 2017). In construction, there is a growing number of studies investigating the possible uses of blockchain to improve digital processes. The literature includes several examples of blockchain applications in fields such as BIM (Turk and Klinc 2017), digital twins (Celik et al. 2021), supply chain (Li et al. 2022) and payments (Luo et al. 2019).

2.3 Blockchain adoption barriers

Investigating blockchain for construction-related purposes is still considered limited from a development standpoint. However, there are several limitations to adopting blockchain in construction-related applications: (1) scalability in public blockchains, whereby public blockchains use consensus to accept transactions on their ledger, limiting its ability to process transactions faster; (2) cost of adoption, whereby the costs for developing blockchain networks and upgrading the current digital infrastructure to support the technology are considered barriers to adopting this technology; (3) legal ambiguity, whereby the use of blockchain is currently not regulated; therefore, regulations are needed to provide legal coverage for the use of blockchain within construction-related activities (Darabseh and Martins 2020).

2.4 Blockchain for intellectual property protection

Blockchain's capability to create an immutable record digitally can provide proof–of-ownership rights for content creators. Digital content is easily replicated, and it is hard to track whether users have respected the rules for use defined by the content owner. Protection of digital content from unauthorised use is the main concern for digital content producers in industries, such as entertainment, literature and software. Examples of studies in which blockchain was used to combat intellectual property infringement follow.

Jing et al. (2021) studied plagiarism in the software development process and the unauthorised use of code.

Their study presented a blockchain-based copyrights management system for combating copyright infringement. Their system used ANother Tool for Language Recognition (ANTLR), a software that identifies code using lexical and syntactic analyses to generate a digital fingerprint and adds it to the blockchain system to find plagiarised content.

Chi et al. (2020) presented a system for book publishing using blockchain to secure content and the purchasers' right to read the book. The system deploys public and private blockchains to create a decentralised book publishing experience, providing a transparent and effective literature management process. Public blockchains store metadata of the literature and purchasing records, which guarantee access to the buyer and royalties to the author. Book content is stored on a private blockchain to ensure that it is only accessible to authorised people who acquire reading rights.

Protecting 3D designs with blockchain is a growing field of research, with solutions that fall into two main classes. The first approach is securing the design transfer process, such as creating a safe channel to access 3D designs with a controlled environment in which designs are protected from unauthorised copying or accessing. Examples of such studies are those by Alkaabi et al. (2020) and Holland et al. (2017), who studied the additive manufacturing process and how to create a blockchain protected design distribution system. The second approach involves certifying design ownership by generating a blockchain-based ownership certificate. Mouris and Tsoutsos (2022) presented a framework for protecting computer-aided design (CAD) 3D designs produced for digital manufacturing by creating a fingerprint of these designs using design spectrogram patterns generated using multidimensional fast Fourier transform and then storing the generated pattern into an NFT to function as the ownership certificate for the design owner.

3 Blockchain design fingerprint as ownership proof for built environment designs

Blockchain's ability to provide an indisputable source of information makes it a perfect fit to provide proof-of-ownership records. To achieve this, a unique collection of identifying information needs to be collected and stored on the blockchain alongside the ownership information. Several storage strategies may be considered, depending on the size of the information. The simplest way to write information is to include it in the transaction input data as a memo, which is suitable for basic uses such as adding the purpose of the transaction. The standard way to store data on a public blockchain is to invoke a smart contract. Smart contracts provide a controlled channel of information to store specific information according to the smart contract rules. In the following sections, a smart contract was deployed to store a built environment design ownership information using an NFT based on the OpenZeppelin (2022) smart contract template.

3.1 Methodology

The study utilises blockchain to tokenise BIM designs as NFTs. The NFT content includes the design ownership information and a multi-dimensional identity. The first dimension of the identity is the global unique identifier (GUID) list, and the second is the IFC entities hash list. The third dimension is the content identifier (CID) list. The NFT metadata is generated based on the ownership information and the generated identity. A standard Ethereum Request for Comment (ERC)-721 smart contract was deployed and invoked to bind the design to an NFT.

3.2 Designing the NFT content

NFT is utilised here to provide proof of ownership for the built environment design based on IFC files. To achieve this, a fingerprint for designs needs to be developed because storing the design itself directly on the blockchain is not feasible, as transaction costs increase according to the content size attached to the transaction. Furthermore, design files have legal liability and defined privacy and sharing rules; therefore, the NFT content should be established considering these factors, using an NFT that merges two groups of information: (1) ownership information; and (2) IFC design file fingerprint. The fingerprint can be defined as a characteristic or an attribute that helps identify its origin. The designed fingerprint has three aspects, as explained in the following sections.

3.2.1 Identifying a design based on the IFC schema specifications

IFC files follow the IFC schema. At the time of writing, IFC 4.0.2.1 is the official version (BuildingSMART 2020a). The schema specifies that IfcRoot is a super-type of all

IFC entities. Each entity that is a subtype of IfcRoot can be defined through a set of identity attributes, which are as follows: (1) GUID (BuildingSMART 2020b), which is a global identifier for the object as defined in International Organisation for Standardisation/International Electrotechnical Commission (ISO/IEC) 11578:1996 (ISO, 1996); (2) owner history, which includes information about the current owner of the object; (3) name, which is a label that reflects the purpose of the object; and (4) description, a set of information that contains comments about the object. Therefore, designs can be referenced according to the GUID for the objects found in the IFC file. The first dimension of the design fingerprint is a list of GUIDs extracted from the IFC file and stored in a separate JavaScript Object Notation (JSON) file using *Ifcopenshell*, a Python library designed to process IFC files. Figure 1 shows an example of GUIDs highlighted within an IFC entity.

3.2.2 Identifying a design file based on entities hash list

The IFC file format is a human-readable data exchange serialisation similar to eXtensible Markup Language (XML) and JSON formats (Wright et al. 2020). This means that the content of the IFC file can be explored as text; therefore, it is possible to perform a cryptographic identification operation on IFC files as text. The IFC file contains three sections: header, body and footer, as shown in Figure 2. The body stores the list of design elements. Each model element is represented as an entity that starts with "# "regardless of the element's geometric or semantic nature.

Sharing the list directly means sharing the full design details; therefore, the second dimension of this fingerprint is a hashed list of the entity list inside an IFC file. In order to generate the hash for each entity, the *hashlib* library (Python Foundation 2022) in Python was used with the parameters defined in Table 1.

SHA256 (Gilbert and Handschuh 2004) was used as a hashing algorithm because, in addition to its collision resistance capabilities, it provides a high level of encryption, resulting in unique hashes when the content is different. Table 2 shows examples of the inputs and outputs of this process.

3.2.3 Identifying a design file based on its InterPlanetary File System (IPFS) CID

InterPlanetary File System (IPFS) is a peer-to-peer file storage system and an essential element for blockchain-based applications as it removes the centralised storage of the application and replaces it with decentralised storage, which contributes to the application's decentralisation robustness (Chen et al. 2017). However, since IPFS is content-addressed, files are accessed according to a unique hash generated based on their content. This identifier, or CID, uses the multihash formula to create self-describing hashes to identify files uniquely (Protocol Labs 2022). The CID is an absolute pointer for the content it represents;

```
#164= IFCSITE('2ER5wG0DXA1fhO9Z$9SMAN',#42,'Default',$,$,#163,$,$,.ELEMENT.,(42,21,31,181945),(-71,-3,-24,-263305),0.,$,$);
```

Fig. 1: Example of GUIDs highlighted within an IFC entity.
GUID, global unique identifier; IFC, Industry Foundation Classes.

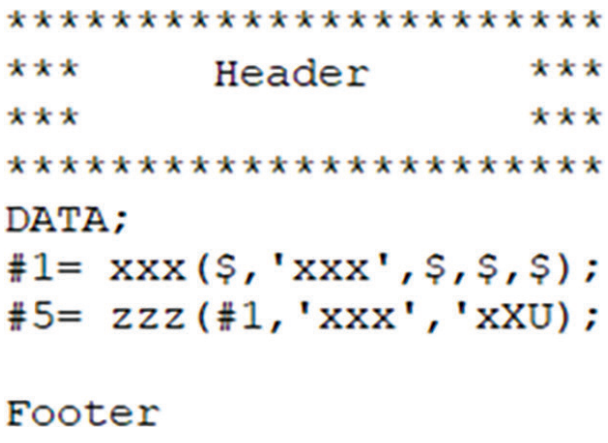

Fig. 2: Illustration of the main sections in a generic IFC file.
IFC, Industry Foundation Classes.

Tab. 1: Hash formula used to generate the entities hash list.

Parameter	Value	Parameter description
Hashing algorithm	SHA256	Determine the desired hashing algorithm
Salt	IFC	Salt is a set of characters used to increase the complexity of the original hashed content
Iteration	1	Number of times that the hash should be rehashed
Length	64	The number of characters in the generated hashes

IFC, Industry Foundation Classes; SHA, Secure Hash Algorithm; SHA, Secure Hash Algorithm.

Tab. 2: Examples of the input and output of the hashing function.

Input (entity)	Output (SHA256 hash)
#12= IFCDIRECTION((1.,0.,0.));	"3d896c6580d86c5895a178d5bdb45ec3901cf3118881495c86caf38b70730a5b"
#14= IFCDIRECTION((-1.,0.,0.));	"fd7a34687b10e217a47657bd93c49340ed00fa59a2f8667a072453160aef9767"
#16= IFCDIRECTION((0.,1.,0.));	"8eeea131de37b45cc0057d63a7e115942257f3315791061adca5ff0704e7f9ea"
#18= IFCDIRECTION((0.,-1.,0.));	"69f375f1c87d560e5981d8168070afdcf6f85953cad3e9d9b6a7e8b218a0c327"

IFC, Industry Foundation Classes; SHA, Secure Hash Algorithm.

therefore, a change in the file will result in a new CID. When a file is added to the IPFS system, it is first processed into data chunks of a defined size (currently 265 KB), and then a CID is generated for each chunk; finally, all the individual chunks' CIDs are hashed to generate the file CID. The CID is a pointer to data, not the data itself. This process generates a Merkle directed acyclic graph (DAG) (Protocol Labs 2020), a common tool to identify cryptographic files, whereby a file can be broken into a defined set of pieces and defined leaf's structure. Each leaf has a hash in the Merkle trees representing the partially hashed file. This approach is commonly used in software development to identify changes and code versioning. The process is illustrated in Figure 3, which shows a DAG for two examples using 32-byte chunks instead of 256 KB, the default chunk size in the IPFS system (Discuss IPFS 2019). In the example, the word "example" was removed from the original text referred to in Figure 3 as Input 1, which resulted in a change in the content of the final 32-byte chunk, which then affected the chunk CID, and as a result, the CID representing the whole file changed when the input text changed.

This concept can be applied to IFC files to generate another dimension for the IFC file fingerprint. Storing the IFC on the IPFS public network will make it accessible to anyone with the IFC CID. Instead, the IFC file can be stored on a private IPFS network to avoid this issue. Users can generate the file CID and sub-CIDs and thereby limit file access to only the members of the private IPFS network. The *ipfshttpclient* Python library (IPFS Shipyard 2021) can be used to add files to the IPFS network and retrieve the file CID sub-CIDs. The library stores the fingerprint dimension files in the IPFS network, which is used later in the NFT generation process.

3.3 Designing and deploying the smart contract

The NFT creation process follows the Ethereum standard ERC-721 (Entriken et al. 2018), which specifies the requirements for NFT smart contracts. An ERC-721 NFT is ownable and transferable. An NFT issued according to the ERC-721 should also have a set of characteristics that make it unique, rendering it identifiable. These characteristics vary depending on the NFT's purpose. Common characteristics of NFTs are as follows: (1) they originate from a smart contract; (2) the smart contract sets the name and symbol, which comprises three or four letters used to identify a smart contract; (3) ID number to internally identify within the smart contract; (4) each NFT has a metadata file accessed through a uniform resource identifier (URI) (Mozilla Developer Network [MDN] 2022). These metadata files are structured in a JSON schema according to the template provided by the ERC-721 standards.

In order to facilitate the smart contract development process and overcome the security concerns of writing smart contracts, the OpenZeppelin (2022) team created a set of smart contracts written in Solidity, which is one of

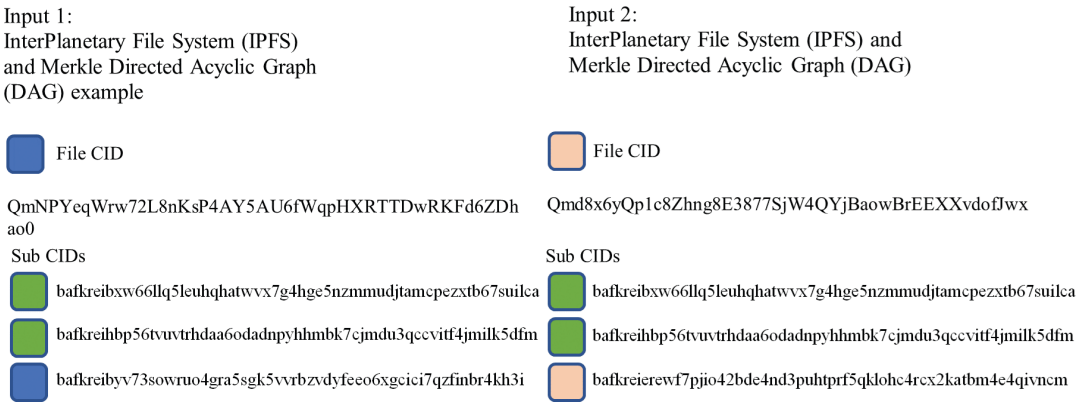

Fig. 3: Example of CID DAG.
CID, content identifier; DAG, directed acyclic graph.

the accepted programming languages for writing smart contracts for the Ethereum public blockchain and other Ethereum virtual machine (EVM)-compatible blockchains. These contracts can be used in two ways: (1) the contract can serve as a template where the developer fills the needed variables and then deploys to blockchain; or (2) a new contract can inherit classes from these contracts. The OpenZeppelin team provides multiple versions of ERC-721-compatible smart contracts to meet the needs of NFTs.

Blockchain development frameworks are another tool used in developing smart contracts and decentralised blockchain applications known as DApps. Frameworks provide a set of tools to facilitate the development process, such as a local testing blockchain and an interface to deploy and interact with the blockchain. Truffle is the most popular framework, which uses JavaScript programming language (ConsenSys 2022); however, in this article, the Brownie framework is used as it is a Python-based framework (Brownie 2020), which is aligned with the remaining parts of the developed code. Brownie provides scripts to interact with smart contracts according to their purpose. Each set of scripts is called a mix (Brownie Mixes 2022). An example of such mixes is the NFT mix. The NFT mix provides advanced and simple NFT deployment options. The simple collectable deployment is a template for deploying a smart contract based on ERC721.sol, which is the standard version of the OpenZeppelin ERC-721-compatible smart contract, however, without using an external data feed. The simple collectable deployment approach is used for this article as the purpose of the smart contract does not require a connection to off-chain information.

It should be stressed that creating the smart contract and creating an NFT are two separate processes, even though one is created to enable the other. The smart contract serves as an NFT factory, as NFTs are created by invoking the smart contract. When using Brownie for creating an NFT smart contract or minting an NFT, two elements are needed: (1) Infura Project ID, where *Infura* (Infura 2022) is an Ethereum application programming interface (API); (2) Ethereum wallet private key, generated using MetaMask (MetaMask 2022) Web wallet. After creating the required information, it is added to the running operating system as environment variables. The next step is defining the NFT name. This name is used to identify the NFT collection and the symbol, which is an abbreviation of the name. In the deployed contract, the NFT name was Building Designs, and the symbol was BDS as the aim of the NFT collection was to prove building design ownership; however, the name could be assigned according to the developer's wishes without restrictions according to the ERC-721 standards. Figure 4 shows the smart contract.

The first lines of the smart contract define the desired compiler version (in this case, 0.6.6 for Solidity), followed by an import statement for the OpenZeppelin ERC-721 smart contract template. The next part of the smart contract is the constructor that initialises the contract, which inherits the ERC721 contract functions imported earlier. The constructor also defines the name and symbol of the NFTs originating from this contract. Furthermore, the constructor maps the number of NFTs generated by the contract. A minting function is defined after the constructor. The first statement defines who is minting the NFT, followed by the URI of the NFT. Then, the number of NFTs is incremented by one before generating the next token.

After developing the smart contract, the next step is to compile it and deploy it on a test network. The smart contract was compiled and developed on the Rinkeby test net network, one of the Ethereum test networks, using the deploying script provided by the Brownie framework.

```
// SPDX-License-Identifier: MIT
pragma solidity 0.6.6;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract SimpleCollectible is ERC721 {
    uint256 public tokenCounter;
    constructor () public ERC721 ("Building Designs", "BDS"){
        tokenCounter = 0;
    }

    function createCollectible(string memory tokenURI) public returns (uint256) {
        uint256 newItemId = tokenCounter;
        _safeMint(msg.sender, newItemId);
        _setTokenURI(newItemId, tokenURI);
        tokenCounter = tokenCounter + 1;
        return newItemId;
    }

}
```

Fig. 4: NFT smart contract written in Solidity.
NFT, non-fungible token.

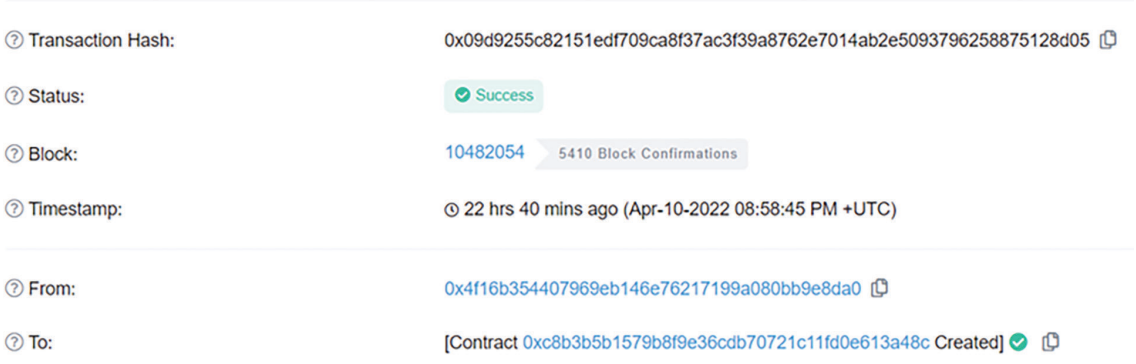

Fig. 5: Transaction confirmation page for deploying the smart contract.
UTC, universal time coordinated.

Figure 5 shows the smart contract deployment transaction on Etherscan.io Ethereum transaction explorer (Etherscan 2022). The transaction page shows the address assigned to the smart contract on the blockchain.

3.4 Invoking the smart contract

The smart contract defines the name of the NFT collection and its symbol. The content of the NFT and what it proves is defined by the metadata of the NFT itself. The metadata URI for each NFT is stored in the blockchain when an NFT is generated. Therefore, the JSON metadata file was designed to accommodate the IFC fingerprint to prove design ownership according to the template in the ERC-721 standard. Figure 6 shows the metadata data example that complies with the JSON schema developed to store the fingerprint filled with information about a sample design.

The schema contains two types of information: (1) basic identifying information, such as the name and description of the design, the author information, the original owner information and attributes for the IFC design file (such as the schema), the original authoring application and the organisation where the design was developed. (2) CIDs as links for files stored on the IPFS storage system, which can be used to identify the design file. The original IFC file is stored on a private IPFS network; however, the IFC identification fingerprint files are hosted on the public IPFS network.

After developing the NFT metadata JSON file and storing it on the public IPFS node, an NFT can be minted by invoking the deployed smart contract. The script of

```
{
  "name": "Design 1",
  "description": "architectural design file use here as an example",
  "image": "ipfs://QmRhgQv7s9Rhgu8J2A2jz3vasbP7dE5ecMhjaqnRpHWNb2?filename=design_1.png",
  "author": "John Doe",
  "original owner": "Jane Doe",
  "original IFC file": "ipfs://Qmd3ygfNT7RGvkpX2wbxEd76JZBFYAGsSSgLxwvdGSvcjC?filename=design_1.ifc",
  "entites hashes list file": "ipfs://QmNeS9qJLwg9CN2UmjP43f4EMZb868CreVx9aRiKB7WGxy?filename=design_1_hashes.json",
  "guids list file": "ipfs://QmdAJU7d5NMw7B4MdgKPYostpNvC9jw1gsjkUsEbAkzwsx?filename=design_1_guids.json",
  "IFC IPFS sub CIDs list file": "ipfs://QmSNqV166NotXGnt1CjfE4zazpvsqHSDhewehwwq9cGwRf?filename=designs_1_cids.json",
  "attributes": [
    { "trait_type": "ifc_schema", "value": "IFC4" },
    { "trait_type": "ifc_setup", "value": "DesignTransferView_V1.0" },
    { "trait_type": "organization", "value": "example_organization" },
    { "trait_type": "authoring_application", "value": "Autodesk_Revit_2022" }
  ]
}
```


Fig. 6: Design information example formatted to comply with JSON schema.
JSON, JavaScript Object Notation.

```python
#!/usr/bin/python3
from brownie import SimpleCollectible, accounts, network, config
from scripts.helpful_scripts import OPENSEA_FORMAT

sample_token_uri = "ipfs://QmWdnfAzJZjwExiEPy8BrrzoGdLsMny8WxxEBec4yASBVE?filename=0-design.json"


def main():
    dev = accounts.add(config["wallets"]["from_key"])
    print(network.show_active())
    simple_collectible = SimpleCollectible[len(SimpleCollectible) - 1]
    token_id = simple_collectible.tokenCounter()
    transaction = simple_collectible.createCollectible(sample_token_uri, {"from": dev})
    transaction.wait(1)
```

Fig. 7: Brownie Python script to mint a design NFT.
NFT, non-fungible token.

Fig. 8: Transaction input data show the URI of the minted NFT.
NFT, non-fungible token; URI, uniform resource identifier.

the Brownie framework to mint an NFT was used after adding the NFT URI to it, as shown in Figure 7. The figure shows that the URI is defined as the NFT URI, and then the minting of the deployed smart contract is called.

After invoking the smart contract, an NFT is generated and assigned an ID by the smart contract. The first NFT gets the ID 0. The transaction details show the IPFS CID for the minted NFT, as shown in Figure 8.

The NFT can be viewed in the OpenSea marketplace (OpenSea 2022) for NFTs to ensure that it is ERC-721 compliant. Figure 9 shows that OpenSea recognises the NFT minted as ERC-721 and the NFT attributes defined in the metadata JSON file.

4 Validating ownership fingerprint

This section presents the research results. The methodology presented in Section 3 was applied in a case study to illustrate how the proposed fingerprint can be used to find similarities between a sample design file and a derivative without accessing the original IFC content.

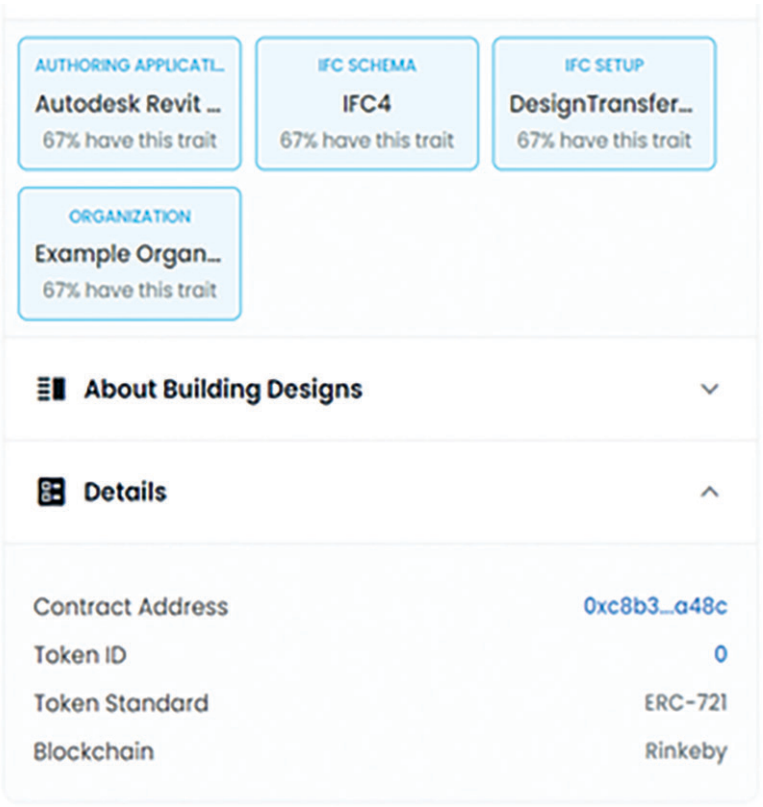

Fig. 9: OpenSea listing for the minted NFT as proof of ERC-721 compatibility.
ERC, Ethereum Request for Comment; NFT, non-fungible token.

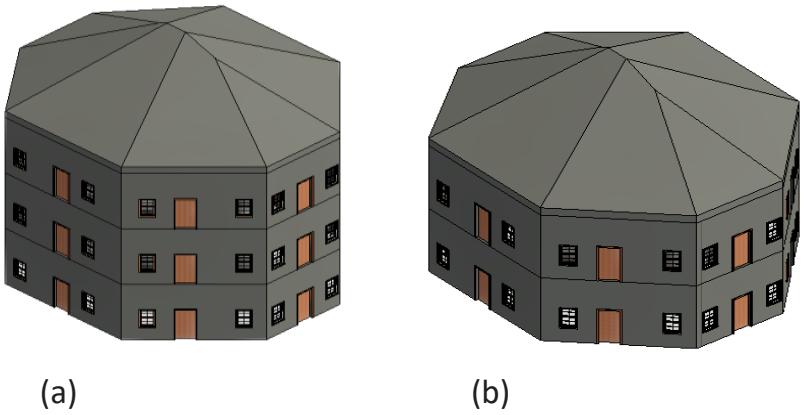

Fig. 10: Test design models: (a) original design; (b) derivative design.

After minting the metadata for both design models, JSON files were retrieved from IPFS and compared. Figure 10 shows the test design models, and Table 3 shows the comparison results.

After retrieving the designs' fingerprint JSON files, they were compared using a match function in Microsoft Excel to identify similarities. The comparison based on the GUIDs found the highest number of similarities, as 238 GUIDs from the original design were also found in the GUID list for the derivative design. The entity hashes comparison found 100 matches between both the design models. This proves that 100 repeated entities are found

Tab. 3: Fingerprint testing results.

Comparison aspects	Original design	Derivative design
Aspect 1: GUIDs		
Number of GUIDs	353	240
Overlapping GUIDs	238	
Aspect 2: Entity hashes		
Number of hashes	7,377	5,232
Overlapping hashes	100	
Aspect 3: IPFS CIDs		
Number of chunk CIDs	3	2
Overlapping CIDs	0	

CID, content identifier; GUID, global unique identifier; IPFS, InterPlanetary File System.

in the derivative design with 100% confidence without acquiring the original IFC files to check explicitly. The third aspect was the chunks of CIDs, which show no similarity. This result was expected because of the small size of the IFC original file compared to the chunk size in IPFS. The original design's IFC file size was 526 KB, and the derivative's file size was 369 KB, compared to the 256 KB chunk size. The number of overlapping GUIDs is higher than the number of hashes because the GUIDs are metadata extracted directly from the IFC file without processing. However, the hash list represents IFC entities digested using an algorithm where a slight change in the entity will result in a new hash. Therefore, the hash list overlaps illustrate the similarity more accurately than the GUID list; however, the GUIDs show unprocessed data that can be found in the original file. The similarity in hashes can be expressed by an absolute similarity index (ASI), which is calculated by dividing the count of overlapping hashes (100 hashes in this example) by the highest number of hashes in the compared files (7,377 in this example). The ASI value ranges between zero and one, where larger values indicate higher similarity (0.0135 in this example). The number of overlapping GUIDs (238 in this example) divided by the higher number of GUIDs in the compared files (353 in this example) yields the relative similarity index (RSI). The RSI can also assume values between zero and one (0.674 in this example).

5 Conclusions

The increased volume of literature on blockchain applications in construction shows the growing interest in this topic. Blockchain can be used to provide an immutable record for digital events, which can be utilised in the construction industry for several purposes, such as ensuring data integrity for Internet of Things (IoT) devices used during the life cycle of built assets or for recording inter-organisational correspondence for use in case of a legal dispute (Darabseh and Martins 2020).

In this article, blockchain was used to create a design file ownership record in the form of an NFT, a standardised procedure used to generate ownership records for digital collectables. The article first presents a novel technique to capture the fingerprint of the building design models encoded as standard IFC files. The fingerprint has three aspects: (1) based on the design's GUID list; (2) based on the IFC entity hash list; and (3) based on the chunk's CIDs generated when the IFC file is stored on an IPFS node. The article also features an ERC-721 smart contract deployment on the Rinkeby Ethereum blockchain and then invokes the smart contract to mint an NFT to function as a design ownership record. The fingerprint was validated by comparing two similar designs.

The ownership of BIM assets is discussed at a full-model level in this article, and similarity indexes are proposed for this purpose. Further work is required to address ownership of BIM classes. Indeed, special intellectual property rights may arise at this level, so classes should be verified independently from a full model check as described in this paper.

Fingerprinting for digital assets instead of the original asset creates a way to validate an asset's content without sharing its full content. However, fingerprinting standards are required to ensure that matching algorithms have defined rules to find similarities and report it.

Further research is required to establish a blockchain standard to tokenise BIM design assets. The ERC-721 standard was not made to work within the construction environment, especially BIM design assets – which are continuously changing, and ownership of information is not always clear.

Funding

The authors acknowledge the financial support from the Portuguese Foundation for Science and Technology (FCT) through PhD grant number 2020.05786.BD. In addition, this work was financially supported by Base Funding - UIDB/04708/2020 of the CONSTRUCT – *Instituto de I&D em Estruturas e Construções* – funded by national funds through the FCT/Ministry of Science, Technology and Higher Education (MCTES) (PIDDAC).

Conflicts of interest

The authors declare no conflict of interest.

References

Adibfar, A., Costin, A., & Issa, R. R. A. (2020). Design copyright in architecture, engineering, and construction industry: Review of history, pitfalls, and lessons learned. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction, 12*(3). doi: 10.1061/(ASCE)LA.1943-4170.0000421.

Alkaabi, N., Salah, K., Jayaraman, R., Arshad, J., & Omar, M. (2020). Blockchain-based traceability and management for additive manufacturing. *IEEE Access, 8*, pp. 188363-188377.

Alreshidi, E., Mourshed, M., & Rezgui, Y. (2017). Factors for effective BIM governance. *Journal of Building Engineering, 10*, pp. 8-01.

Beach, T., Petri, I., Rezgui, Y., & Rana, O. (2017). Management of collaborative BIM data by federating distributed BIM models. *Journal of Computing in Civil Engineering, 31*, p. 04017009.

Brownie. (2020). *Brownie — Brownie 1.18.1 Documentation*. Available at https://eth-brownie.readthedocs.io/en/stable/ [accessed 13 April, 2022].

Brownie Mixes. (2022). *Brownie mixes*. GitHub. Available at https://github.com/brownie-mix [accessed 13 April, 2022].

BuildingSMART. (2020a). *IFC4 Documentation*. Available at https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/ [accessed 13 April, 2022].

BuildingSMART. (2020b). *IfcGloballyUniqueId. Globally Unique Id*. Available at https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC2/HTML/schema/ifcutilityresource/lexical/ifcgloballyuniqueid.htm [accessed 13 April, 2022].

Celik, Y., Petri, I., & Rezgui, Y. (2021). Leveraging BIM and blockchain for digital twins. In: *2021 IEEE International Conference on Engineering, Technology and Innovation, ICE/ITMC 2021 - Proceedings*. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICE/ITMC52061.2021.9570246.

Chan, D. W. M., Olawumi, T. O., & Ho, A. M. L. (2019). Perceived benefits of and barriers to building information modelling (BIM) implementation in construction: The case of Hong Kong. *Journal of Building Engineering, 25*, p. 100764. doi: 10.1016/j.jobe.2019.100764.

Chen, Y., Li, H., Li, K., & Zhang, J. (2017). An improved P2P file system scheme based on IPFS and Blockchain. In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 2652-2657.

Chi, J., Lee, J., Kim, N., Choi, J., & Park, S. (2020). Secure and reliable blockchain-based eBook transaction system for self-published eBook trading. *PloS one, 15*, p. e0228418.

ConsenSys. (2022). *Truffle | Overview – Truffle Suite*. Available at https://trufflesuite.com/docs/truffle/[accessed 13 April, 2022].

Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation, 2*, p. 71.

Darabseh, M., & Martins, J. P. (2021). Protecting BIM design intellectual property with blockchain: Review and framework. In: *Proceedings of the 38th International Conference of CIB W78, Presented at the CIB W78 2021*, Luxembourg.

Darabseh, M., & Martins, J. P. (2020). Risks and opportunities for reforming construction with blockchain: Bibliometric study. *Civil Engineering Journal (Iran), 6*, pp. 1204-1217. doi: 10.28991/cej-2020-03091541.

Discuss IPFS. (2019). Does IPFS provide block-level file copying feature? - Lobby. discuss.ipfs.io. Available at https://discuss.ipfs.io/t/does-ipfs-provide-block-level-file-copying-feature/6388 [accessed 13 April, 2022].

Entriken, W., Shirley, D., Evans, J., & Sachs, N. (2018). EIP-721: Non-fungible token standard. *Ethereum Improvement Proposals*. Available at https://eips.ethereum.org/EIPS/eip-721 [accessed 13 April, 2022].

Etherscan. (2022). *About us*. Available at https://etherscan.io/aboutus [accessed 13 April, 2022].

Gilbert, H., & Handschuh, H. (2004). Security analysis of SHA-256 and sisters. In: Matsui, M., & Zuccherato, R. J. (eds.), *Selected Areas in Cryptography, Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, pp. 175-193. doi: 10.1007/978-3-540-24654-1_13.

Holland, M., Nigischer, & C., Stjepandić, J. (2017). Copyright protection in additive manufacturing with blockchain approach. In: *Transdisciplinary Engineering: A Paradigm Shift*. IOS Press, pp. 914-921.

Infura. (2022). *Welcome to Infura Docs*. Available at https://docs.infura.io/infura [accessed 13 April, 2022].

IPFS Shipyard. (2021). Available at py-ipfs-http-client/files.py at master ipfs-shipyard/py-ipfs-http-client.

ISO. (1996). ISO/IEC 11578:1996. ISO. Available at https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/00/22/2229.html [accessed 13 April, 2022].

Jing, N., Liu, Q., & Sugumaran, V. (2021). A blockchain-based code copyright management system. *Information Processing & Management, 58*, p. 102518.

Kugler, L. (2021). Non-fungible tokens and the future of art. *Communications of the ACM, 64*, pp. 19-20.

Kuo, T. T., Kim, H. E., & Ohno-Machado, L. (2017). Blockchain distributed ledger technologies for biomedical and health care applications. *Journal of the American Medical Informatics Association, 24*, pp. 1211-1220.

Li, X., Lu, W., Xue, F., Wu, L., Zhao, R., Lou, J., et al. (2022). Blockchain-enabled IoT-BIM platform for supply chain management in modular construction. *Journal of Construction Engineering and Management, 148*. doi: 10.1061/(ASCE)CO.1943-7862.0002229.

Luo, H., Das, M., Wang, J., & Cheng, J. C. P. (2019). Construction payment automation through smart contract-based blockchain framework. In: M, A.-H. (ed.), *Proceedings of the 36th International Symposium on Automation and Robotics in Construction, ISARC 2019*. International Association for Automation and Robotics in Construction (I.A.A.R.C), pp. 1254-1260. doi: 10.22260/isarc2019/0168

MDN. (2022). *Identifying resources on the Web - HTTP | MDN*. Available at https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Identifying_resources_on_the_Web [accessed 13 April, 2022].

MetaMask. (2022). *Introduction | MetaMask Docs*. Available at https://docs.metamask.io/guide/ [accessed 13 April, 2022].

Mouris, D., & Tsoutsos, N. G. (2022). NFTs for 3D models: Sustaining ownership in industry 4.0. *IEEE Consumer Electronics Magazine*.

Ølnes, S., Ubacht, J., & Janssen, M. (2017). Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. *Government Information Quarterly*, *34*(3), pp. 355-364. doi: 10.1016/j.giq.2017.09.007.

OpenSea. (2022). *Getting Started – OpenSea*. Available at https://support.opensea.io/hc/en-us/sections/360011539774-Getting-Started [accessed 13 April, 2022].

OpenZeppelin. (2022). *OpenZeppelin Contracts*. Available at https://github.com/OpenZeppelin/openzeppelin-contracts [accessed 13 April, 2022].

Protocol Labs. (2022). *Content Addressing | IPFS Docs*. Available at.https://docs.ipfs.io/concepts/content-addressing/#identifier-formats [accessed 13 April, 2022].

Protocol Labs. (2020). *Merkle DAGs | IPFS Docs*. Available at https://docs.ipfs.io/concepts/merkle-dag/ [accessed 13 April, 2022].

Python Foundation. (2022). *hashlib — Secure Hashes and Message Digests — Python 3.10.4 Documentation*. Available at https://docs.python.org/3/library/hashlib.html [accessed 13 April, 2022].

Sardroud, J. M., Mehdizadehtavasani, M., Khorramabadi, A., & Ranjbardar, A. (2018). Barriers analysis to effective implementation of BIM in the construction industry. In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. IAARC Publications. pp. 1-8.

Tapscott, A., & Tapscott, D. (2017). How blockchain is changing finance. *Harvard Business Review, 1*, pp. 2-5.

Turk, Ž., & Klinc, R. (2017). Potentials of blockchain technology for construction management. *Procedia Engineering, 196*, pp. 638-645. doi: 10.1016/j.proeng.2017.08.052.

Wright, A., Andrews, H., Hutton, B., & Dennis, G. (2020). *JSON Schema: A Media Type for Describing JSON Documents*. Available at https://json-schema.org/draft/2020-12/json-schema-core.html [accessed 13 April, 2022].