Blockchain asset lifecycle management for visual content tracking

Alexandre C Moreaux, Mihai P Mitrea

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 July 2023, accepted 28 August 2023, date of publication 4 September 2023, date of current version 20 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3311635

RESEARCH ARTICLE

Blockchain Asset Lifecycle Management for Visual Content Tracking

ALEXANDRE C. MOREAUX AND MIHAI P. MITREA
Samovar Laboratory, Télécom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France
Corresponding author: Mihai P. Mitrea (mihai.mitrea@telecom-sudparis.eu)

ABSTRACT With the current complexification of image manipulation technologies (ranging from color editing or aspect ratio modifications to AI-generated fake news), many numerical representations can be connected to the same semantic visual content. Thus, ensuring trust and authenticity for tracking near-duplicated visual content (*i.e.*, semantically identical yet digitally different content) becomes challenging from both methodological and technical points of view. Addressing these challenges requires the synergistic combination of methodological solutions from different research fields, while current solutions are heterogeneous and lack interoperability. In this paper, we create an automatic complete lifecycle management workflow for visual content assets represented on blockchains. The workflow is supported by a novel architecture seamlessly integrating near-duplicated content detection, Smart Contract automation, and token brokerage. The architecture leverages a load-balancing framework and near-duplicated content detection to grant properties natively featured by blockchains (security, trust, and transparency) to the authentication of assets in environments where the same semantic content has various digital representations. Subsequently, minted blockchain assets can be used contingently with other state-of-the-art tools, ensuring interoperability with blockchain working standards. The effectiveness of this workflow is demonstrated through open-source example implementations for the Ethereum and Tezos frameworks, illustrating the benefits this process brings to automatic asset generation and Intellectual Property Rights (IPR) management.

INDEX TERMS Asset management, blockchains, decentralized applications, load management, smart contracts, tokenization, visual information retrieval.

I. INTRODUCTION

Visual content is one of the highly valuable assets on the market today. From entertainment movies [1] and social networks [2] to video content generated by uncrewed vehicles [3] or by AI [4], nearly every applicative vertical benefits from advancements in digital content products and services. The global digital creation market was valued at close to 26 billion USD in 2022 and is forecasted to reach 70 billion USD before 2030 [5]. While multimedia assets represent a foundation of the modern digital economy, they can still suffer from improper use or IPR (Intellectual Property Rights) infringements, such as copying, illicit commercial exploitation, resource waste, or false appropriation [6]. The online piracy market was estimated at 51.6 billion USD in 2022 [7], and abuses have an immediate, tangible influence on fields such as the movie industry [8]. Moreover, multimedia content must face novel issues in web3[1] environments [12], such as metaverse content creation [13], [14], which raises environmental [15], economic [16], and social [17] challenges to be tackled by innovative solutions from both public and private sectors.

Consequently, no matter the specific application and throughout its entire lifecycle, visual content protection is currently achieved by a large variety of conventional solutions, ranging from *data encryption* (ensuring privacy in data transmission and storage [18]), *digital signatures* (tracking

The associate editor coordinating the review of this manuscript and approving it for publication was Shu Xiao.

[1] Web3 refers to the paradigm shift of the web from the social application-oriented interactive era (web2) to the open decentralized era focused on Distributed Ledger Technologies, cryptocurrencies, Artificial Intelligence, etc. [9], [10], [11].

the content by compact digests of its digital representation [19]), *watermarking* (tracking the content by inserting additional data inside it [20]), to *digital fingerprinting* (tracking the very semantics[2] of the content by compact digests of its human perceived features [22]).

In this paper, **we focus on the authentication of digital assets using visual fingerprinting**. This methodological framework produces a semantically invariant digest out of the perceptual characteristics of a given visual input, thus allowing visual content to be tracked via its semantic features. Concisely, visual fingerprinting is designed to reach a trade-off between its **unicity** (*i.e.,* semantically different contents shall result in different fingerprints) and **robustness** (*i.e.,* semantically identical yet digitally different contents shall result in similar fingerprints). As such, slightly modified versions of the original content can also be matched back to their original. **While this methodological framework is compatible with web2 environments, it does not feature any inner security properties, which can be found in web3 environments.**

Currently, decentralized environments [23], blockchains at the forefront, are commonly known not only as a support for transactional systems but also as generic solutions for ensuring immutable, zero-trust security[3] [24] for a large variety of digital assets. These assets range from generic cryptocurrencies [26] to unique decentralized assets with a variety of purposes [27], [28], [29], passing by standard web2 [30] or physical assets [31], [32]. Although limited in their computation capabilities and storage availability, blockchains innovated the paradigms of a wide range of conventional businesses, including finance [33], auditing [34], and IPR management [35], to mention but a few. **Hence, our study investigates whether and how blockchains can complement visual fingerprinting solutions by providing them with security and resilience features.**

Coupling blockchains with visual content fingerprinting presents neither conceptual nor theoretical contradictions. However, the association between the two is drastically restricted by the lack of methodological bridges and by critical limitations in their technical and functional properties. First, a disconnect in the processing workflow appears between visual content assets and their blockchain representations. Not only does the level of web3 abstraction need to be accounted for in a trustworthy manner, but the same semantic content, although unambiguously identified by human beings, presents a potentially infinite number of digital representations that can be processed. Secondly, visual content processing is prohibitively complex to be executed on blockchain environments (further referred to as *on-chain* processing, as opposed to legacy *off-chain* processing). Accommodating computationally intensive operations intrinsic to visual content processing, distribution, and storage in an environment so heavily constrained in resource usage presents a significant challenge to our subject. Thirdly, current-day implementation efforts are typically limited to specific use cases, which leads to the absence of widely adopted interoperable standards as of the time of writing. **As such, we aim to create and automate a systematic process for managing decentralized assets representing physical or web2 assets created *on-chain* making the most out of modern visual content processing.**

To this end, we build on and extend ideas brought forth in [36], [37], and [38] to provide automatic, end-to-end semantic content tracking through authenticated *on-chain* digital assets that are unambiguously and persistently linked to *off-chain* visual content assets. Specifically, **we conceive, design, implement, and evaluate an *on-chain*-*off-chain* load-balancing architecture that accommodates: (1) *off-chain* data storages; (2) semantic content identification through fingerprinting techniques; (3) Smart Contracts for immutable storage; (4) MPEG-21 Part 23 Smart Contract prototyping for format standardization; (5) *on-chain* assets, including their protection and distribution.**

The resulting framework enables multimedia content to be connected to decentralized assets while bearing the trace of an authentication process, guaranteeing its semantic unicity. Such assets can then be upgraded with blockchain protection functionalities such as royalty management tools, handed to state-of-the-art *on-chain* brokers, and distributed accordingly. Once out in the environment, these assets carry the trace of this authentication process and can be used with no restrictions, thus ensuring backward compatibility with existing solutions.

While targeting the automation of a comprehensive, end-to-end workflow catering to the needs of creation and distribution of *on-chain* assets representing *off-chain* visual content assets, the main contributions of this paper are:

- An architecture enabling the systematic generation of authenticated assets as well as their subsequent management, featuring:
  - A mechanism establishing mutually beneficial associations of *on* and *off-chain* applications for said authentication,
  - An easy to operate automated workflow, with built-in protection against mistakes,
  - Backwards and forwards interoperability with contingent state-of-the-art solutions.
- An open-source software implementation of the above architecture for the Ethereum framework, alongside key modules for the Tezos framework. This software is shown to function standalone and as part of an already established workflow.

The present paper is structured as follows. Section II is devoted to state-of-the-art concepts and tools for fingerprinting, applicative blockchains, and their joint usage. Section III

[2]Throughout this study, the semantic content of visual assets is defined as the sum of their perceptual characteristics as understood by a human being [21].

[3]We use the notion of zero trust formalized in [24], *i.e.*, the assumption that there is no implicit trust granted to assets or user accounts based solely on their physical or network location, nor based on asset ownership. This notion is often summarized by "*never trust, always verify*" [25].

advances the methodological pillars of our solution before combining them into a novel architecture. In Section IV, we detail the software implementations of the architectural components, while end-to-end proof-of-concept implementations are instantiated for two use cases in Section V. Section VI analyses the paper's contributions, highlighting their strengths and identifying their limitations. Section VII concludes and lays out future work.

II. STATE-OF-THE-ART

This section begins by providing a global view of visual content tracking (Section II-A) and applicative blockchains (Section II-B) before investigating their joint uses (Section II-C). Its findings (summarized in Section II-D) identify the current-day deadlocks, thus serving as a base for the design principles we identify and adopt for our methodology (Section III).

A. FINGERPRINTING-BASED VISUAL CONTENT TRACKING

As previously mentioned, visual content protection can take different approaches. Although the workflow we will present could benefit from data encryption at the level of data storage or hidden signatures without any extra drawbacks, it focuses on content tracking. Indeed, while turning multimedia content into web3 assets, said content naturally becomes accessible and hence vulnerable to copying or abusive usage. As such, the first step is to enable the tracking of content by identifying it.

Cryptographic hashing functions [39] might seem like a natural solution to identify content through fixed-length digests, allowing one to look for copies throughout data storages efficiently. However, multimedia content scarcely remains strictly similar when being processed. Even omitting malicious modifications, changes naturally occur in file format, encoding, and metadata when stored, sent, or used. These changes give birth to near copies, or near duplicate content, of the same original multimedia excerpt. If one were to take a picture using a camera, upload it to their computer, and send the picture to someone else, three instances of the picture would exist. Each of these instances would be different from the others in digital representation and, as such, would have different cryptographic hashes. However, we would commonly call all three the ''same picture.'' Formally, [40] provides four definitions related to near-duplicated video content. The most useful to our paper is the most general, namely: *"Identical or approximately identical videos close to the exact duplicate of each other, but different in file formats, encoding parameters, photometric variations (color, lighting changes), editing operations (caption, logo, and border insertion), different lengths, and certain modifications (frames add/remove)."*

In opposition to cryptographic hashing, near-duplicated content tracking (also referred to as visual fingerprinting) is a methodological framework that considers the semantic content of its input when creating the digest. As such, it can identify slightly modified versions of multimedia content, *i.e.,* near-duplicated content. Consequently, one can check if two fingerprints are strictly equal and how similar they are, as illustrated in Figure 1.

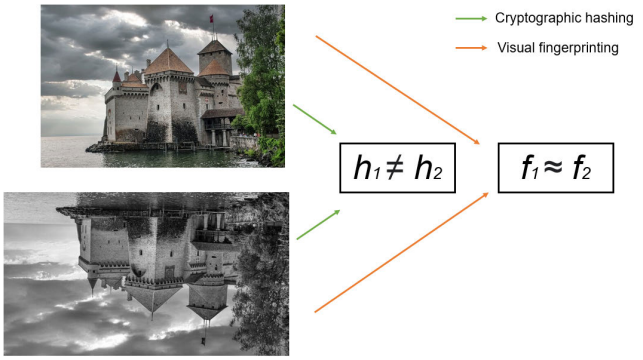


FIGURE 1. Illustration of the differences between digests produced by cryptographic hashes and similarity preserving visual fingerprints.

Given this feature, visual fingerprints can be used in advanced feature-based comparisons [41], using a variety of multimedia formats as inputs (*e.g.,* audio files) [42]. Nevertheless, their applications are not limited to feature detection and can range from Digital Rights Management [43] to network protection [44]. Their primary use remains video identification, for which various methods have emerged. [45] presents a survey of the landscape of fingerprinting for video files while [46] benchmarks performances of widely used methods. Performance requirements also affect the specifics of given methods; some only need to compare sparse inputs, while others are expected to detect near copies in web repositories [47]. Visual fingerprinting has also repeatedly been associated with deep and machine learning to be trained and optimized over specific sets of data [48], [49], [50], [51]. Some fingerprinting methods allow for various input formats, while some include metadata and instance data in their methodology.

The broader characteristics of fingerprinting methods also diverge significantly. For instance, the format of the digest identifying the semantic content can range from short strings to large matrices. Targeted performances can explain these differences, *i.e.,* semantic information required to identify an input with the desired robustness. Once a method is decided upon, a similarity threshold is selected to reflect the chosen near-copy detection sensitivity. A low sensitivity will consider mildly different inputs as unique, while a high sensitivity will flag minor differences more precisely. The robustness relates to the capacity to identify near-duplicated content after it has been modified by standard attacks (*i.e.,* visual content modifications, be they malicious or not). For instance, while some fingerprints might not perform well after the resizing of an input, others might suffer from changes in luminosity or color balance.

The output format also defines the comparison methods that can be used to verify similarities between inputs. While the Hamming distance or the bit error rate can be used for

binary digests, the most recurrent methods revolve around using normalized correlation functions. Detection thresholds are to be adjusted depending on the level of similarity needing to be detected.

As the introduction mentions, fingerprinting methods do not feature native security features and can be modified when recorded in standard data storage. We aim for blockchain to provide additional security and data integrity to state-of-the-art fingerprinting methods. **The architecture presented in this paper will use a combination of cryptographic hashing and visual fingerprinting. Cryptographic hashing will allow us to turn large fingerprints into smaller digests, while fingerprinting technology will power the near-duplicated content detection aspect of the process.**

B. APPLICATIVE BLOCKCHAINS

Blockchains are peer-to-peer anonymous networks of nodes that produce a sequence of cryptographically linked blocks. These blocks contain information about the transactions that have occurred in that network. Blockchains function as a zero-trust third party in the exchange of assets and information [52]. They now support a wide array of applicative verticals [53], [54], reflecting the heterogeneity of the modern blockchain ecosystem [55].

1) SMART CONTRACTS

The fundamental brick of application-oriented blockchains is the Smart Contract [56], an automated piece of code written in blockchain-specific language (*e.g.,* Solidity for Ethereum or Michelson for Tezos), which runs as programmed, with no possibility of change or influence from any central authority or unauthorized party. Smart Contracts provide tamper-proof, zero-trust features, and complete data integrity but suffer from low processing capabilities and lack of maintenance, which bars them from competing with their web2 counterparts [57]. Furthermore, blockchain programming languages do not support as many features as *off-chain* ones (*e.g.,* Python or Java), limiting the complexity of the data formats and the operations that can be included in a Smart Contract.

Specifically, all operations in blockchains (*e.g.,* transactions, Smart Contract deployments) require gas, *i.e.,* a fee to compensate for the power used on the blockchain. This gas is spent by validators elected via a consensus algorithm [58] to include transactions in a block. Hence, all operations in a Smart Contract consume a certain amount of gas, making heavy operations expensive or even impossible due to the gas limit of the blocks. The nature of the execution of operations in blockchain environments also entails that they take significantly longer to be processed than in web2 environments, as even the most basic transaction must be validated and included in a block before it is executed.

Smart Contract security, programming language limitations, and performance problems are identified by [59] as significant challenges developers face when tackling applicative blockchain environments. In practice, these issues materialize in the risk of unmodifiable mistakes (bolstered by code being public and hence targetable), the lack of general-purpose libraries, the absence of support, or the constrained number of local variables. As such, it is not only best practice but mandatory to keep Smart Contracts simple and intuitive to avoid them becoming prohibitively expensive or significant liabilities. Our study deals with the secure storage of these Smart Contracts, and although the specific strategy is novel, we do not add any novel aspect to data storage. Numerous studies have innovated unique features, including repair groups [60] and large-scale computability [61].

2) TOKENS AND THEIR DISTRIBUTIONS

Another central notion of blockchains is the token, *i.e.,* the blockchain representation of digital assets. Tokens are either native transactional tokens used to compensate for computing power and pay for services on the blockchain or custom tokens created by users according to widely accepted blockchain-specific application-level working standards [62], [63], [64]. User-created tokens can be fungible (interchangeable and splitable, as per legal tender) or non-fungible (representing unique assets and being undividable). Both native and customs tokens can be owned and traded by users and Smart Contracts.

Application-level standards are typically set by the community and serve as trusted precedents for best practices. Token formats are such standards and function as secure and open ledgers for digital assets to be accounted for, wielding supply management tools alongside secure transfer functions. For instance, the Ethereum community creates ERCs – Ethereum Request for Comments standards [65] that allow anyone to submit ideas that have fostered community support: the ERC20 [66] standard corresponds to Fungible Tokens (FTs), and ERC721 [67] to Non-Fungible Tokens (NFTs). Hybrid standards also exist, notably the multi-token ERC1155 [68].

NFTs are sometimes used as representations of digital art in a massive market where decentralized exchanges recorded a trading volume of 343 billion USD in the second quarter of 2021 [69]. Specifically, blockchain marketplaces are decentralized applications that provide vast arrays of features and rules but fundamentally connect NFT sellers and buyers for a fee. Most rely on a Swap Contract model, where a central Smart Contract manages sales, as illustrated in Figure 2 [70].

Interestingly, interoperability between marketplaces is limited, even when they operate in the same blockchain ecosystem. Moreover, NFTs have suffered repeated reputational damages due to malicious actors [71], [72], [73], with controversial news making their way to mainstream, non-specialized media outlets [74]. Accomplished careers and online fame have even been achieved by investigating NFT schemes and frauds [75], [76]. Even the most prominent actors have acknowledged that NFT abuse is rampant. Namely, *Opensea* (a historic Ethereum marketplace that,

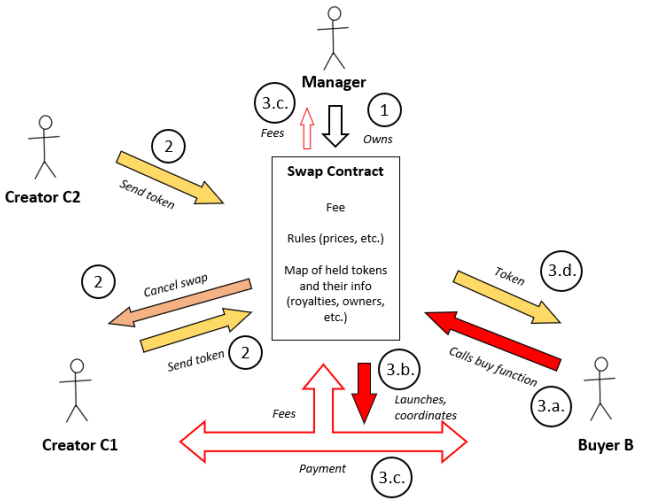


FIGURE 2. Simplified Swap Contract workflow [70]. Numbers represent steps, in order; yellow arrows show token movement; red arrows show purchase execution; white and red arrows show royalty distribution.

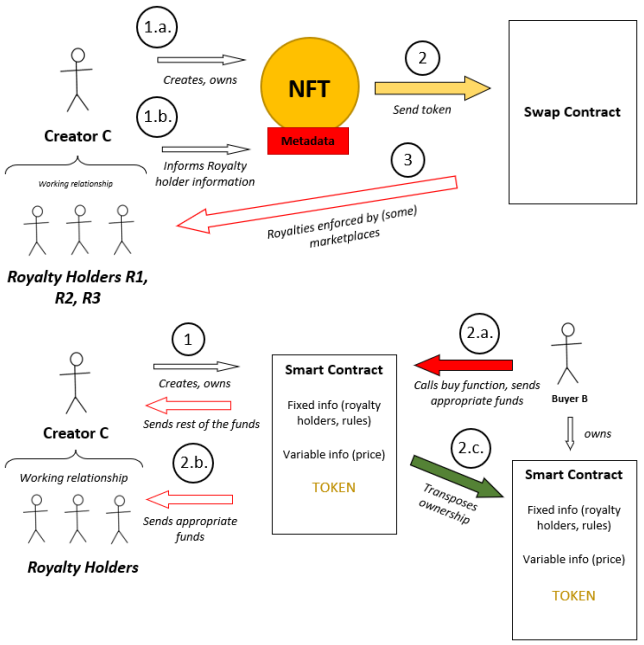


FIGURE 3. (Top) EIP2981 distribution workflow; (bottom) Royalty Management Token-Level Smart Contract (RM-TLSC) from [70]. Numbers represent steps, in order; white arrows show ownership; red arrows show purchase execution; white and red arrows show royalty distribution; green arrows show changing of hands of the RM-TLSC.

according to [77], led the space and continues to be a frontrunner) had to change its simplified creation process because it was vastly contributing to plagiarized works, fake collections, and spam [78]. The protection of multimedia-based blockchain assets as a whole and the importance of their relationship with the financial market was investigated in a report [79] issued by the Organisation for Economic Cooperation and Development (OECD), which put the protection of these assets into perspective. Further technological opportunities associated with tokenizing novel assets are studied in [80].

Although the most common use for NFTs is digital art, they can enable more complex processes that could require or benefit from different distribution methods. For instance, IPR management can be traced by exchanging designated tokens. In this respect, [81] used NFTs to manage patented and copyrighted IP traceability. Yet, distributing these tokens in open marketplaces is not always the best solution. Other use cases also require content to be monitored or subjected to specific limitations (*e.g.,* royalties). Consequently, some of these use cases use complementary state-of-the-art distribution solutions (*e.g.,* marketplaces), while others require a more dedicated means of distribution.

3) TOKEN PROTECTION

Tokens being at the center of the blockchain economy, their protection is of high stakes. This fact is reinforced when the tokens represent real-world assets, as discussed in this paper. While the security of the asset within the blockchain environment is assured by native blockchain mechanisms and careful private key management, abuse regarding the relationship with the original asset is not guaranteed. Notably, ownership and IPR issues are at risk in blockchain environments, exacerbated by the fact that a sizeable portion of purchasers and potential purchasers do not fully understand what they possess after acquiring an NFT. Buyers obtain two distinct pieces of information when acquiring a token:

- A digital token that contains metadata that often points to *off-chain* storage.
- A usage license [82].

These licenses are usually very restrictive and limited to creative commons and personal use licenses [83]. As such, proper token protection worries itself with the specifics of the license it provides for future owners. However, even assuming these points are correctly understood and applied, further protection cannot be ensured without complementary solutions.

Notably, another prominent issue with tokens lies in the ability to distribute royalties alongside sales. Stances on royalties are one of the major differentiating factors of marketplaces, which must position themselves with respect to limits for the primary and secondary markets. These royalties are far from negligible, as Ethereum marketplace royalties have already added up to almost 2 billion USD [84]. Marketplace-agnostic royalty modules also exist and can be added to our token before being distributed, *e.g.,* the EIP2981 [84] standard on Ethereum or a Royalty Managing Token Level Smart Contract (RM-TLSC) brought forth in [70]. The former relies on Intellectual Property (IP) and royalty metadata being appended to the token, which sellers can then enforce, while the latter sets the token in a separate Smart Contract, which enforces a set of rules systematically. Figure 3 shows simplified schemas of the initial distribution lifecycle of assets implementing both solutions.

The issues and limitations above invite one to consider applying advanced visual content protection techniques to these assets. Unfortunately, blockchains cannot support the required computational load for the fingerprinting methods presented in Section II-A. Nevertheless, the robustness and immutability of blockchains are features we target to combine with fingerprinting methods. **As such, we aim to create a framework enabling modern content-tracking techniques to benefit from blockchain properties.**

C. DIGITAL FINGERPRINTING AND BLOCKCHAINS

The association of visual fingerprinting and applicative blockchains makes sense on a conceptual level. Indeed, content protection technologies such as visual fingerprinting provide advanced solutions in identifying and detecting sometimes modified content. At the same time, blockchains can bring a layer of security through the immutability of their open ledger. However, their combination is limited by computational capacities and the lack of operational bridges. **In this paper, we focus not only on combining tools present in both on and *off-chain* environments but on providing an architecture that allows the mutually beneficial association of said tools.**

This association, although scarce, has previously appeared in the scientific literature. As mentioned in Section I, this paper extends a previous study [37] that incorporated visual fingerprinting elements in blockchain environments. As such, its state-of-the-art is also relevant to this paper. Briefly, blockchains can aid visual content tracking as part of the process [86] or alongside *off-chain* processing [87]. Encryption, watermarking, and transaction tracking fingerprinting are notably reviewed in [12]. To our knowledge, only [37] combines visual fingerprinting near-copy detection with blockchain. The proof of concept for the *on-chain* - *off-chain* connection we use in our workflow was brought forth in [88], in which the authors advance a lightweight framework for document tracking. This idea was extended in [89] to include a blockchain shadow database to track original inputs, which we also make the best of later in this paper (*c.f.,* Section III). Other combined uses of multimedia technologies and blockchains have notably been studied for IoT [90] and cloud computing [91]. To the best of our knowledge, multimedia technologies have only been associated with blockchains as partial support to an overall process, and the lifecycles of physical or web2 assets (*e.g.,* visual content) and their web3 counterparts have been kept separate. In this study, we aim to ensure the symbiosis of both paradigms for the entire life cycle of web3 assets, from their creation to their eventual distribution.

D. SUMMARY

This section investigates the state-of-the-art of visual content tracking, blockchain applications, and assets, as well as their joint uses. It brings to light that:

- Visual fingerprinting can be a powerful tool in semantic content tracking for various use cases (*e.g.,* archive protecting, IPR tagging). However, its applicative scope is limited by the lack of native security mechanisms.
- Blockchain features trust and immutability that would benefit visual content tracking but cannot support the computation requirements set by fingerprinting operations.
- Coupling both technologies is conceptually and theoretically possible but drastically limited because of the lack of methodological bridges, their antagonistic technical properties, and their divergent functionalities.
- Current solutions are targeted at specific use cases and require the manual processing of parts of their workflow, in addition to lacking support across standards and environments.

In this paper, we create a systematic, interoperable workflow that can provide automatic and complete lifecycle content identification. The following section will detail our methodology and its components on a conceptual level before we consider practical illustrations and complete implementations further in the paper.

III. METHODOLOGY

This section describes the advanced architecture and workflow alongside the specifics of the on and *off-chain* methodological bricks that constitute them. We explain our approach to the architecture design (Section III-A) before describing it and our workflow in general terms (Section III-B). Then (Section III-C), we detail how the individual components of the architecture contribute to the workflow before ending (Section III-D) with a macro description of the process. The implementation details are left for presentation in Section IV and for illustration in Section V.

A. ARCHITECTURE DESIGN

The advanced solution reconsiders, extends, and innovates state-of-the-art studies to automate the entire lifecycle of multimedia assets into blockchain environments so they can serve a broad spectrum of further uses.

Our solution will follow an architectural design conceived to overcome the current-day limitations identified in Section II. Specifically, the workflow accommodated by the architecture shall comply with the following five design criteria (DC):

- **DC1.** It shall support assets throughout their entire life cycle.
- **DC2.** It shall identify the semantic content of an input and not the digital representation of the input *per se*.
- **DC3.** It shall identify the content in a robust fashion.
- **DC4.** It shall be backward and forward compatible, *i.e.*, it shall be non-intrusive regarding complementary state-of-the-art solutions while being open to further use.
- **DC5.** It shall be versatile in supporting of diverse input formats, blockchain environments, and token standards.

Figure 4 illustrates the main successive processing steps assets undergo during our process. The specifics of this workflow are detailed hereafter.

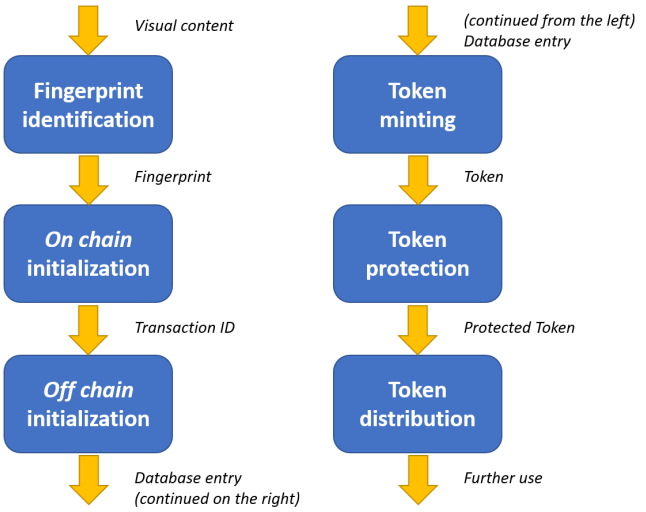


FIGURE 4. General workflow of our solution, starting with a piece of visual content and ending with a distributed token. The left side shows semantic comparison processing, and the right side shows token related actions. The two are sequential in practice.

Let us start with a set of web2 visual content (*e.g.,* images, video) whose semantic content we want to create a trusted precedent for, as per **DC1**. We designed this solution to enable extra content to be appended to the set anytime. We begin by identifying semantic features of our content for comparison purposes using a fingerprinting method, as shown in the first block on the left side of Figure 4 and detailed in Section II-A, to fulfil **DC2**. Simply storing the fingerprints would make them vulnerable, static, and unfit for content distribution. This approach would contradict **DC3** and **DC4** above. Consequently, we initialize this information on the blockchain to benefit from the environment's immutable data tracing and comply with **DC3** before storing the fingerprints. Storage resource limitations naturally invite us to hash the information before recording it *on-chain*. This operation corresponds to the second block on the left side of Figure 4. Once certain the information appears in a Smart Contract, we store the initial fingerprints in an *off-chain* data storage, as shown in the third and last block of the left side of Figure 4. This redundancy will enable advanced checks we explain in Section III-C. When further candidate entries go through the same process, they are verified for their semantic unicity against all previously verified entries.

At this point of the process, we have an immutable trace of semantic data processing, establishing a precedent for the original content. To enable the further use of the assets with contingent state-of-the-art solutions (as dictated by our **DC4**), we create a standardized token (as presented in Section II-B2), which includes links to the data storage entries. This operation is the first block on the right side of Figure 4. This token can either represent the original or a related asset, *e.g.,* an asset representing exploitation rights or IPRs. We then protect this token with state-of-the-art solutions (as presented in Section II-B3) and hand them to a Token Broker for distribution. These steps are illustrated in the second and third blocks of the right column of Figure 4, respectively. As such, the asset bears the immutable trace of the semantic verification for the rest of its life cycle. This process intrinsically makes provisions for a variety of inputs as well as for the simultaneous use of various blockchain environments and token standards, as per **DC5**.

B. ARCHITECTURE DESCRIPTION

The advanced architecture, illustrated in Figure 5, answers the five requirements discussed above and ensures seamless data processing between on and *off-chain* technological bricks. Specifically:

- The *App*: The central piece of the architecture. It processes the multimedia content, communicates with the other blocks, and serves as the interface with the operator; details are presented in Section III-C1.
- The *Smart Contract*: The *on-chain* piece of code serving as an unfalsifiable integrity check for the *off-chain* database; details are presented in Section III-C1.
- The *off*-chain database(s): The lightweight databases holding the hashes of the fingerprints for the recorded content. The advanced architecture is agnostic concerning the database technology; details are presented in Section III-C1.
- The *Token prototype*: A piece of code that uniformizes the input format of the information processed to create (or *mint*) the tokens; details are presented in Section III-C2.
- The *Token Contract(s)*: One or several Smart Contract(s) that establish(es) the capacities and limitations of the tokens while also generating them on demand in a permissioned fashion; details are presented in Section III-C2.
- The *Token Protection Module*: A Smart Contract or piece of code in a standard Smart Contract that adds a layer of protection to the tokens before they are distributed. This step should be customized according to the specific use case; details are presented in Section III-C2.
- The *Token Broker*: A state-of-the-art Smart Contract or decentralized application (DApp) that connects token buyers and sellers; details are presented in Section III-C2.

We will elaborate on each one of these elements in the following sections. The operating workflow of the architecture is performed in two steps:

- A qualified blockchain expert accomplishes the initial setup of the database and blockchain Smart Contracts.
- Once set up, an unqualified operator can use the architecture, only ever needing to interact with the App.

The process starts with visual content being fed to the App and fingerprinted. The generated fingerprints are then checked for near copies in the *off-chain* database. This copy-detection process, illustrated in Figure 6, can lead to three results:

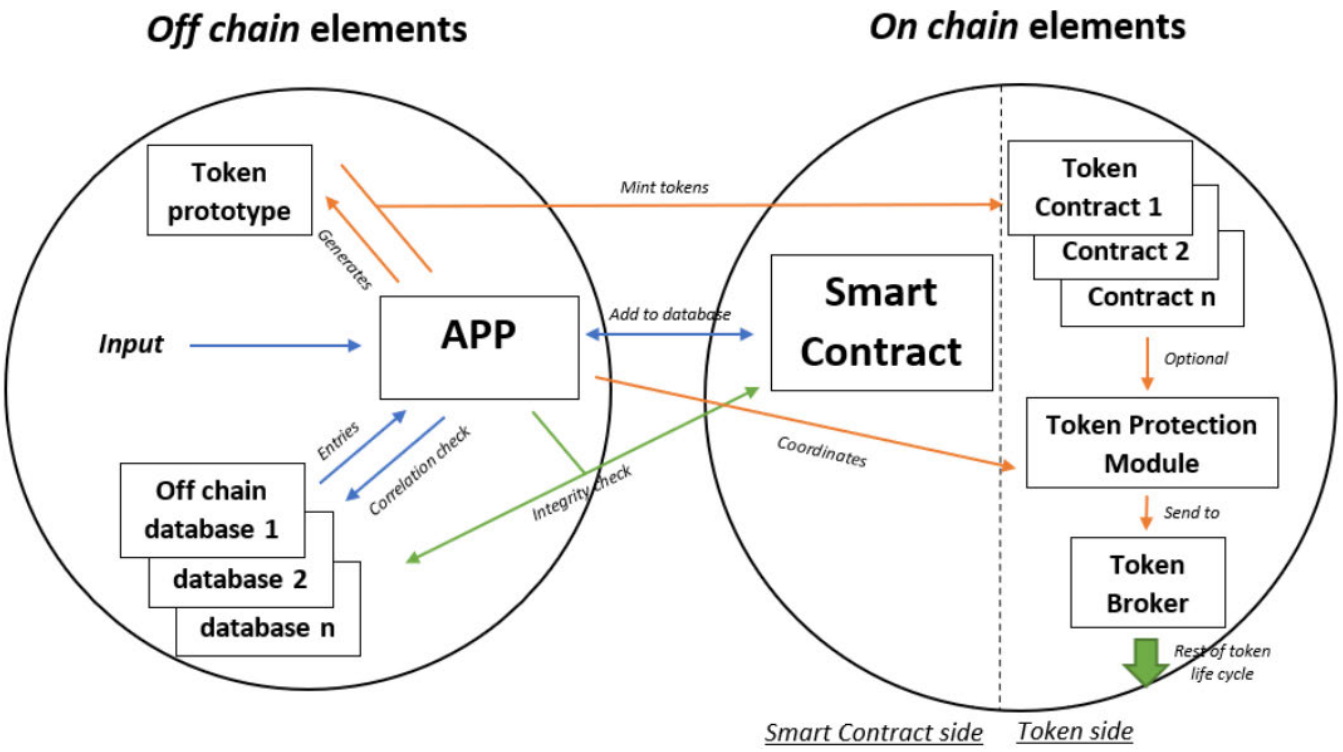


FIGURE 5. Advanced architecture. The modules are grouped according to their environment of execution.

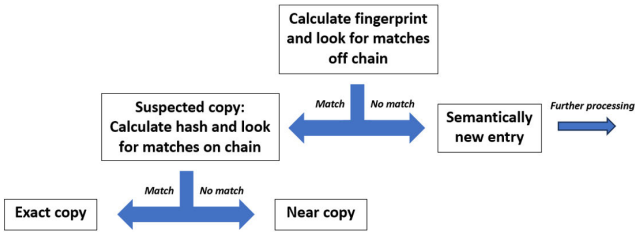


FIGURE 6. Multimedia semantic content comparison flowchart.

- The input is detected as a copy of existing content (*i.e.,* the fingerprint is identical to an entry of the database), the operator is informed as such, and the process stops.
- The input is detected as a near copy of one of the entries according to the designating threshold (cf. Section IV). The operator is informed as such, and the process stops.
- The content is not detected as a copy or near-copy of any existing entry. The content can be added to the database.

Suppose the input is considered original in semantic terms (according to the matching criterion of the fingerprinting method). In that case, it can be initialized on the blockchain, specifically in the Smart Contract's storage. This storage serves as a pseudo database that shadows the *off-chain* database. This tamperproof (because *on-chain*), redundant database allows the Smart Contract to serve as an arbiter, ensuring the database has not been tampered with. Once the App has received confirmation of this operation, it adds the original input to the *off-chain* database. These operations are illustrated in the complete architecture shown in Figure 5.

Upon arriving in the *on-chain* database, the entry is minted as a Non-Fungible Token according to the Token Prototype, optionally protected with complementary solutions, and sent to a Token Broker. This token could also be sent to a designated wallet per the use case.

C. FUNCTIONAL BRICKS

1) NEAR-DUPLICATED CONTENT DETECTION (APP, SMART CONTRACT,*Off-chain* DATABASES)

The near-duplicated content detection of this architecture uses the database, Smart Contract, and App. It is powered by fingerprinting technology and enables the authentication of the information eventually used in the tokens.

As mentioned, the database only holds the fingerprints of the recorded content. Although it would be possible to hold the content in the database and fingerprint it upon retrieval, a lighter database that does not hold any content directly allows for faster processing and fewer potential privacy concerns. The database must only pass given fingerprints to the App on request.

The Smart Contract is used on two occasions: to provide information to the App during the greenlight function to cross-check the database entries (explained in the following two paragraphs) and to process a new entry admissible in the database. The former does not require input data, while the latter requires a hash and an optional string of general

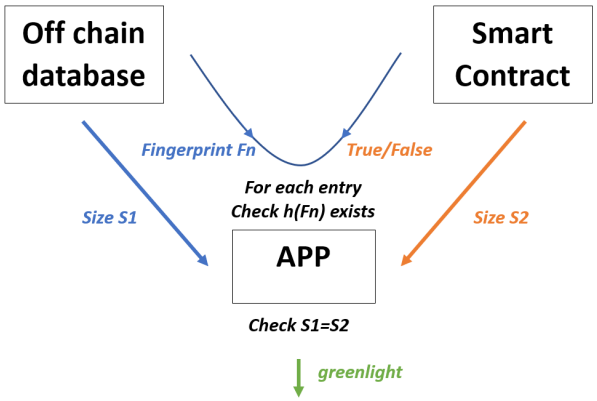


FIGURE 7. Illustration of the greenlight function's database verification, which unlocks the rest of the process.

information recorded with the entry. Both are mapped to a Boolean, indicating their existence.

The App interacts with both databases and is also the only point of contact for the operator. As such, the visual interface can be designed to make the process intuitive and easy to operate. In the context of an academic study, we did not develop any graphical interface and interacted with the App using a command prompt. The App is given a multimedia file (whose format is dictated by the fingerprint in context) as an input parameter. It begins by establishing a connection with the Smart Contract. The greenlight function is immediately called prior to any operation. This function returns True, allowing the process to continue, if and only if the *off-chain* and *on-chain* databases match. It does so by retrieving the size of the map of hashes and using the compare function of the Smart Contract. As such, the App ensures that each database entry appears *on-chain* and that no other entries do. This process is expedited because the database contains fingerprints that need not be reprocessed systematically. This process is shown in Figure 7.

The greenlight function returning False immediately interrupts the process and informs the operator that the databases have been tampered with. Assuming this important control passed, the App calculates the input file's fingerprint and compares it to all the entries in the *off-chain* database. Because of the previous greenlight check, this processing can be performed entirely *off-chain*, enabling this workflow's computational efficiency. As explained in Section III-B, three results can be given to the operator: **copy**, **near copy**, or **no copy**. In the latter case, the operator may prompt the App to add the input to the database. If this is done, the App transactions the Smart Contract via the deployer wallet to add the hash of the new fingerprint to the Smart Contract and the *off-chain* database. Note that the fingerprint is hashed before being stored in the Smart Contract because of format and storage concerns in blockchain environments (*e.g.,* matrices are not supported). If the fingerprint in context happens to output short identifiers (*e.g.,* the International Standard Content Code [91] considers four different 72-bit strings), the hashing step may be skipped as it is not essential to the proper

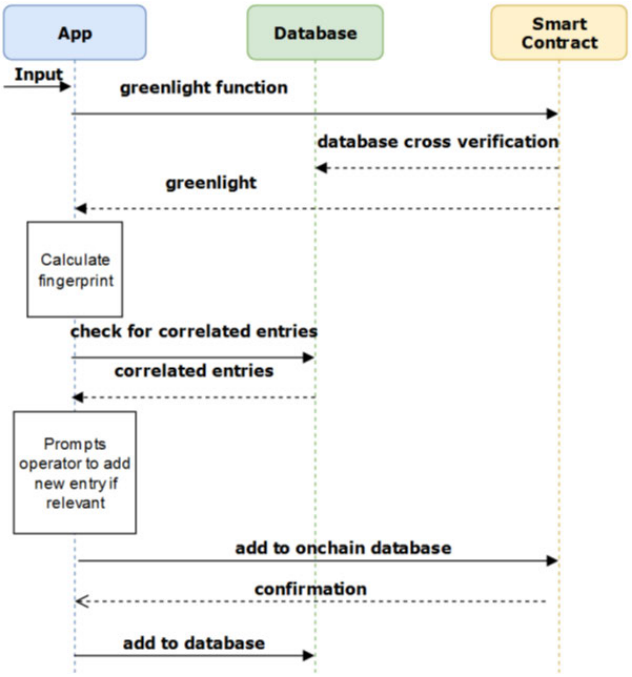


FIGURE 8. Sequence diagram of the step-by-step addition of a new entry in the databases.

functioning of the code, although it adds a layer of privacy to the information. These successive states are illustrated through a sequence diagram in Figure 8.

This paper also introduces the possibility of simultaneously accommodating multiple databases and tokens. Indeed, use cases might require categorizing content according to features such as format or origin. Given the lightweight approach of the databases, a single database can support multiple types of inputs on the simple condition that it can filter out specific sets to send to the App. This concern is rendered quasi-trivial by modern data storage technologies. A single Token Contract can be used if an appropriate mixed standard exists (*e.g.,* ERC1155 [68]). The only required adaptations for the App and Smart Contract are separate variables representing the distinct databases. The architecture's flexibility enables its scalability: it can be expanded to a vast realm of use cases while only being minimally impacted in terms of computational requirements and gas cost. Such considerations would not be practical if they required the multiplication of every methodological brick in the architecture.

2) TOKEN MANAGEMENT (TOKEN PROTOTYPE, TOKEN CONTRACT, TOKEN PROTECTION MODULE, TOKEN BROKER)

We now present the various complementary aspects of the token lifecycle: creation, definition, protection, and distribution.

During the initial setup, a Token Contract is deployed on the blockchain. This Token Contract follows a standard explained in Section II (*e.g.,* ERC721). Note that the Token Contracts need not be deployed on the same

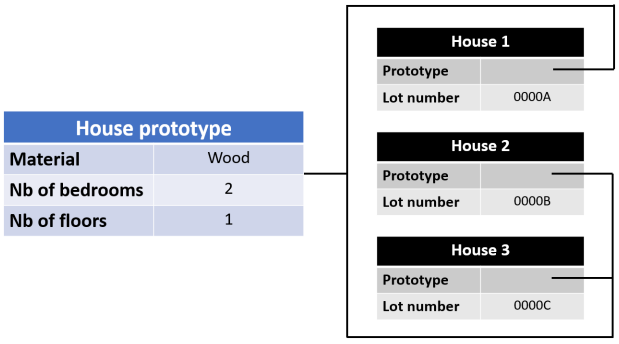


FIGURE 9. A basic example of prototyping. Data formats are made simpler by the presence of a standard unifying element.

blockchain on which the Smart Contract is deployed. The tokens can be minted by multiple standards on any (or multiple) blockchain(s). Once deployed, the Token Contracts can be called individually to create blockchain-specific assets on demand or simultaneously to create equivalent assets on different blockchains. The presence or absence of advanced features of the standards in the Token Contracts is imposed by the specifics of a given use case. The only hard requirement is a mint function, which the App will call to generate tokens. This function is called once the transaction that adds a new entry to the Smart Contract is executed and confirmed.

To uniformly identify the assets we create, we use a systematic way to generate them. To do so, we take inspiration from code Prototyping [93], a form of instance-based programming that establishes the reuse of existing objects, sometimes across different platforms of programming languages. Prototypes are dictionaries of variables and functions that enable the translation of code from one environment to another in a systematic fashion. For instance, [94] uses a prototype to generate Smart Contracts from MPEG-21 standard CEL/MCO IPR specifications [95]. Figure 9 illustrates a basic example of prototyping by standardizing all but one characteristic of a set of houses. In our case, we do not need complex indexing functions but the simple formatting of input variables to be passed to various Token Contract constructors. This way, the traces of the near-duplication detection are ensured to be uniform across assets and platforms.

The creation process of the assets and architecture are also adaptable to the simultaneous use of multiple token standards. The only added cost in this expansion comes in the initial deployment of Token Contracts. In use, the App will only call the relevant Token Contract(s) upon adding a new entry to the database. The gas cost of the initial deployment evolves linearly with the number of Token Contracts to deploy. Unfortunately, no state-of-the-art mechanism can curb these costs at the time of writing. On the plus side, our solution makes the Token Contract the only technological brick that must be multiplied when using this architecture for simultaneous input formats and token standards.

Once the token is minted, it can optionally be protected (*e.g.,* in terms of IP or royalties) with one of the state-of-the-art tools explained in Section II and illustrated in Figure 3. For instance, our software makes provisions for adding royalty-related metadata through EIP2981 or the enforced protection of IP rights via an RM-TLSC. For the former, the IERC2981 interface would be implemented and signaled on the ERC721 token through the `_registerInterface` function and equipped with the appropriate royalty information for supported marketplaces. More ample details can be found in [85]. For the latter, an RM-TLSC Smart Contract Prototype (such as [96]) would be taken and customized via its initialization function, which allows to set rules that will be automatically enforced in a zero-trust fashion. These rules can, for instance, relate to price caps, royalty payments, or IP limitations. This initialization will also send the token to the Smart Contract, where it remains until retraction rules are fulfilled according to the initializer's demands. A simple instance of such a Smart Contract and its use can be found in [97], and further details about TLSCs can be found in [70].

The output of these methods can be a token or a Smart Contract, which can then be distributed appropriately with the necessities of a use case. This distribution can be as simple as sending the asset to the deployer wallet (or any other designated wallet) for safekeeping and further use, as standard as sending the asset to a marketplace, or as complex as tailor-made solutions to distribute the asset in a permissioned fashion. We explore the former in Section V-A and the latter in Section V-B, in an example where an automatic broker manages the sale of access FTs, which can be exchanged for the permissioned access of live content provided by an IoMT Camera.

D. SUMMARY

This section presents the consecutive steps taken in the processing of an input. We assume the input is a piece of visual content semantically original with regards to the rest of the database inputs and that it will pass the fingerprint comparison successfully:

- Step 1: The *on-chain* and *off-chain* fingerprint storages are compared. If they match, the process is greenlit.
- Step 2: The input is fingerprinted. From this point onward, the original content is no longer used.
- Step 3: The fingerprint is compared to the entries in the *off-chain* database. It is detected as a copy, near-copy, or no copy. The process continues in case of no copy.
- Step 4: The fingerprint is added to the Smart Contract, which prompts its addition to the *off-chain* database.
- Step 5: The App generates the token input using the Token Prototype and the specific input information.
- Step 6: The Token Contract is called and mints a unique NFT.
- Step 7: The NFT is empowered with the required security resources and passed on to the desired Token Broker.

These steps are illustrated in a simplified sequence diagram in Figure 10.

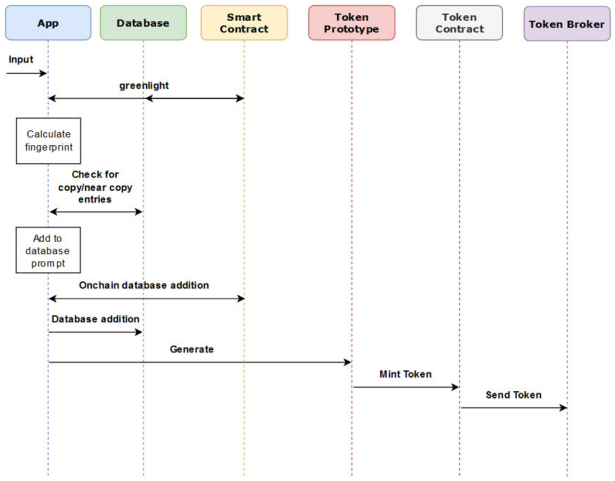


FIGURE 10. Sequence diagram of the step-by-step addition of a new entry in the database.

This framework allows for the creation of digital assets representing semantically verified content. These digital assets bear the trace of this verification yet are compatible with various solutions in the blockchain space and can be used with no added restrictions. The following section will show the technological bricks for nondescript use cases before we apply our workflow to real-world scenarios.

IV. IMPLEMENTATION DETAILS

In this section, we focus on the technical details of the methodological bricks of the architecture. While some are standard and can be used throughout the interfaces provided by their creators, some are not and require a level of detail to understand how and why they were built as such. Hereafter, we shall go over both, referring to the standard tools we used and detailing those that are not. More space shall be allotted to methodological bricks who innovate with regard to current-day standards and those with a more significant role in the architecture.

Specifically, we first go in-depth on the Smart Contract and illustrate its functioning through an example developed for the Tezos infrastructure because of the simplicity of the sandbox environment and the visualization tools provided by Tezos' SmartPy [98] (legacy code can be run on [99]). Then, we will discuss the appropriate token standards and complementary solutions for our architecture for the Ethereum and Tezos blockchains. Throughout the section, we will alternatively illustrate *on-chain* tools for both Ethereum and Tezos to highlight that our architecture is deployable throughout multiple blockchain infrastructures. Finally, the last part of this section will refer to the technologies used for the *off-chain* elements of our architecture, highlighting their features and potential replacements.

Access control not being a central feature of this paper, we set out two simple parties: (1) `creator`, which initializes the Smart Contract and has all access rights, and (2) `otherUser`, which only has the basic view right default to all blockchain users and cannot modify the Smart Contract.

TABLE 1. Environment parameters used in the Tezos smart contract.

Blockchain	Tezos
Environment	SmartPy
Language and version	SmartPy v0.16.0

A. THE SMART CONTRACT

Before we showcase complete implementations in Section V, we show the basics of the Smart Contract's format and use with Tezos's SmartPy [98], [99] and a generic test scenario executed in a sandbox environment. The familiar syntax of the language should enable readers unfamiliar with *on-chain* development to understand the role of the Smart Contract. Please note that Tezos development is based on meta-programming; as such, the code we write is not a Smart Contract but serves to construct the actual Smart Contract that will run on the blockchain [100]. The settings used in this section are summarized in Table 1.

The Smart Contract implements five functions. Three of these functions are of `get` type and allow communicating information about the *on-chain* storage to the App. They return the size of the map, the information associated with a hash, and the Boolean associated with a hash, respectively. The latter serves as the `compare` function called by the App during the greenlight function. The other two functions manage database entries, respectively, providing the `addition` and `deletion` of entries. The `addition` function verifies the prior inexistence of the entry in the database, maps the new entry, and returns a Boolean upon processing. The `deletion` function checks for the entry's existence and adjusts the map's size if needed before returning a Boolean.

Please note that within this proof-of-concept, the deletion (or *burning*) of the token created alongside the entry in the database does not occur. The `addition` and `deletion` functions can only be called by the address that deployed the Smart Contract. If a use case requires multiple addresses to call the Smart Contract, an allowlist can replace the ''only deployer'' approach. If multiple databases and Token Contracts were required, the Smart Contract would simply bear a new map of entries for each one. The functions would then use an extra input parameter specifying the map the App wishes to access and modify.

Figure 11 shows an implementation of this Smart Contract on SmartPy. The functions are straightforward and lightweight, as required by blockchain development. We voluntarily did not include the constructor of the Smart Contract to focus on the functions explained above.

The first two functions shown in Figure 8 will manipulate the map of hashes sent by the App in the complete workflow and are, as such, tagged *@entry_point*. The three later functions shown in Figure 11 are the get functions, which pass on information to the App. They only read data and are hence

```
@sp.entry_point
def addToDB(self, info, hash):
    sp.verify(sp.sender==self.data.owner, 'Only owner.')
    sp.verify(self.data.fingerprintHashes.contains(hash)==False)
    self.data.fingerprintHashes[hash] = sp.record(info=info)


@sp.entry_point()
def deleteEntry(self, hash):
    sp.verify(sp.sender==self.data.owner, 'Only owner.')
    del self.data.fingerprintHashes[hash]

@sp.onchain_view()
def compare(self, hash):
    sp.result(self.data.fingerprintHashes.contains(hash))

@sp.onchain_view()
def getSize(self):
    sp.result(sp.len(self.data.fingerprintHashes))

@sp.onchain_view()
def getInfo(self,hash):
    sp.result(self.data.fingerprintHashes[hash].info)
```

FIGURE 11. A SmartPy implementation of the architecture's Smart Contract.

preceded by *@onchain_view*. Now, we build an illustrative test scenario for this Smart Contract. It is shown in Figure 12. This scenario is powered by SmartPy's Smart Contract test features.

We begin by setting up the scenario and the `creator` and `otherUser` accounts. We then attempt to initialize the *on-chain* storage as the App would, illustrated in the second block of transactions of Figure 11. When creator requests the addition of a new, adequately formatted input, the transactions are executed without issues, as done in the first line of Figure 11's second bloc. On the contrary, inputting a previously recorded entry and `otherUser`'s attempts are rightfully unallowed operations and are hence denied by the blockchain (or *reverted*). At the term of these operations, the Smart Contract's storage matches what is shown in Figure 13.

The addition of new inputs can be prone to errors, as illustrated in the third set of transactions of Figure 11, where an unwanted entry is added before being deleted by `creator`, while `otherUser` is shown to be unable to affect said entry. At the term of these transactions, the storage is back to the state shown in Figure 13. We consider this storage to correspond to the initial storage of a given use case.

When the App executes the greenlight function, the operations that constitute the last bloc of transactions shown in Figure 11 are requested. The App starts by checking that the number of hashes recorded in the Smart Contract equals the number of entries it counts in its local storage (3 in this case) and then uses the compare function to check that said hashes match. If this test passes, the greenlight function returns `True` to signal that the *off-chain* and *on-chain* databases match, enabling the rest of the workflow.

The Smart Contract used in the Section is available in SmartPy's online IDE via [101]. The test scenario can be run using the "Run Code" button at the top left of the screen.

```
scenario = sp.test_scenario()
creator = sp.test_account("creator").address
otherUser = sp.test_account("otherUser").address
c1 = FingerprintStorage(owner=creator)
scenario += c1

c1.addToDB(hash="0x0001", info="first input").run(sender=creator)
c1.addToDB(hash="0x0001", info="same input").run(sender=creator, valid=False)
c1.addToDB(hash="0x0002", info="unauthorized user").run(sender=otherUser, valid=False)
c1.addToDB(hash="0x0002", info="second input").run(sender=creator)
c1.addToDB(hash="0x0003", info="third input").run(sender=creator)
scenario.show(c1.getInfo("0x0003"))

c1.addToDB(hash="0x9999", info="input including a mistake").run(sender=creator)
c1.deleteEntry("0x9999").run(sender=otherUser, valid=False)
c1.deleteEntry("0x9999").run(sender=creator)

scenario.verify(c1.getSize() == 3)
scenario.show(c1.compare("0x0001"))
scenario.show(c1.compare("0x0002"))
scenario.show(c1.compare("0x0003"))
```

FIGURE 12. An illustrative SmartPy test scenario that manipulates the Smart Contract as the App would in our workflow.

FingerprintHashes		Owner
Key	Info	tz1aRxdK2bTSg...
'0x0001'	'first input'	
'0x0002'	'second input'	
'0x0003'	'third input'	

FIGURE 13. SmartPy Smart Contract storage after the addition of the three inputs shown in Figure 9.

The Smart Contract is also deployed to the Ghostnet Tezos testnet at the following address:

KT1MSVoHdoYQpWPBXw4QbfvjSU1abizJbzM2.

Its functions, storage, and deployment figures can be searched using a Tezos explorer, such as TzKT [102].

B. TOKEN ELEMENTS

While the Smart Contract had to be constructed to meet the demands of the architecture, the *on-chain* elements related to tokens are much more standard.

First, the Token Contract must follow a working standard for security and acceptance purposes. Various standards exist on each blockchain infrastructure to accommodate various use cases—the Smart Contract presented in Section IV-A would likely be used alongside the FA2 standard [103]. Ethereum users could consider ERC721 [67] for purely NFT support or ERC1155 [68] for multi-token support. An implementation of these standards can be found at [104] and [105], respectively. Token Contracts can be generated online via intuitive wizards, such as [105]. The specifics or the recorded information will appear via the Uniform Resource Identifier (URI) field, which points to the metadata of the token instance.

Once an entry is added to *on* and *off-chain* storages, the Token Contract mints a new token on demand of the App. This token can be protected via tools such as EIP2981 or an RM-TLSC. Once the token has been equipped with the desired protection, it can be sent to standard or specific

distribution methods, according to the use case. The tools for the token protection and distribution steps are explained in Section III, and we illustrate examples in Section V.

C. OFF-CHAIN ELEMENTS

As explained in Section III, the database holds fingerprints where the App can retrieve them. Lightweight implementations can satisfy themselves of local file storage thanks to redundancy and greenlight checks brought by the blockchain and the App, respectively. We did as such for the implementations in Section V.

Although a more complex Token Prototype can be necessary when multiple blockchains host various tokens simultaneously, it remains a simple element in single-platform uses. For instance, simple lines of code unifying the software format can be used. In our case, the Token Prototype merges into the App in the architecture presented in Figure 3. For instance, using the ERC721 standard, we can pass on the deployer wallet information alongside the fingerprint hash. This hash is recorded in the URI of the token, which stores its metadata. In Section V, we opted to populate said URI with the hashed fingerprint of the original input. This data could be made more thorough, but its specifics, once again, depend on the use case.

If this architecture were used to certify content before it is sold as original, one could imagine additional information being present in the token to ensure the good standing of the token's underlying content. Such additional information may relate to the transaction number of the initial admission of the entry in the database, the address of the Smart Contract where the fingerprint is recorded, the threshold used during the authentication process, or the electronic signature of the issuing body operating the database.

Providing a unique, one-size-fits-all solution for any of the flexible bricks would hamper this workflow's broad spectrum of potential applications.

Finally, we built the App using Python and the web3py [106] library to interact with the blockchain. Any programming language that interacts with databases and blockchains can be used similarly. The complete App we used is available at [108] and was only modified to accommodate various fingerprints tailored to the various use cases. We allowed the near-copy threshold to be set during initialization for testing purposes, but it would remain fixed for complete implementations.

D. SUMMARY

In this section, we provided illustrative examples of the methodological bricks presented in Section III. Specifically, we detail the format and functionalities of a generic Tezos Smart Contract that could be used in our methodology before illustrating the variety of working standards and complementary state-of-the-art solutions that can seamlessly be slotted into our general architecture.

TABLE 2. Environment parameters used in the Ethereum implementations.

Blockchain	Ethereum
Environment	Private EEA Hyperledger Besu
Language and version	Solidity 0.8.18
Token standard	ERC721

Consequently, we put each methodological brick in perspective with the complete workflow presented in Section III, explaining why we specified them and how they contribute to said workflow. We also used this discussion to point out the flexibility of various components to suit specific needs in real-world contexts. We shall continue supporting this point in the next section, where we assemble these technologies into our complete architecture, which we put at the service of two real-world use cases.

V. USE CASE ILLUSTRATIONS

This section shows how our solution can benefit two visual content management use cases. First, Section V-A presents how our architecture can serve as the backbone to a use case simulating a museum being wary of the IP of their online content. Then, Section V-B demonstrates how our architecture and workflow can be flexibly integrated into other advanced solutions by automating and adding content traceability functionalities to an IoMT use case dealing with monetizing live video content.

Both are implemented on a 3-node, Hyperledger Besu EEA (Enterprise Ethereum Alliance [108])-compliant Proof of Authority private blockchain deployed on an Amazon Web Services server, as well as on the now deprecated Rinkeby Ethereum testnet[4] accessed through the Infura node cluster; these characteristics are shown in Table 2.

A. MUSEUM VISUAL CONTENT TRACKING

In this use case, we implement the architecture presented in Section III and Figure 5 without integrating other high-level operations. It is a pragmatic use case rooted in a simplified workflow to be found in [37]. The database belongs to a museum that wants to protect the content they put online from being copied and redistributed fraudulently. We decided to apply this scenario to the virtual visit of six rooms offered by the Louvre Museum in Paris during the COVID-19 pandemic [110]. We use these images for strictly academic and non-commercial purposes and do not intend any infringement of the Louvre's IPR. We sampled images of the visit of these six rooms on keyframes containing semantic content to be protected (*e.g.,* paintings) in sequences of one frame per second. Separate rooms were treated as different inputs.

[4]The specifics of our testing setup do not affect the software presented in the paper, which is compatible with EVM (Ethereum Virtual Machine) [109] blockchains.

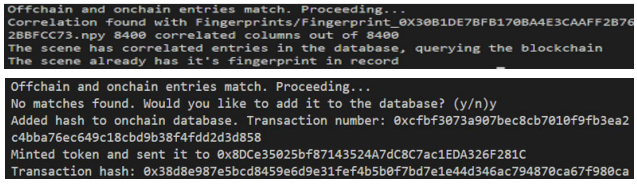


FIGURE 14. The App's output when processing a copy (top), and an original input which is added to the database (bottom).

We elected to use a robust video fingerprinting method brought forth in [111]. This method is optimized for live recordings and invariant regarding scale and affine transformations, which will be part of our tests. We compared these fingerprints using a normalized correlation method with a threshold decided upon for general detection purposes of 0.7. In this use case, near-copies cannot be added to the data storage. Our open-source software implementation for this use case is available at [108]. It contains the App, Smart Contracts, Token Contracts (for the ERC721 standard), Token Prototypes, and lightweight local storage of the inputs described above. The Smart and Token Contracts were deployed on our local architecture (detailed in the introductory paragraph of this section) at the addresses stored in the eponymous folder.

We initialized the six original rooms to the database and Smart Contract. To evaluate our setup, we generated modified versions of the inputs (*i.e.,* near copies) and unrelated content composed of other images from the virtual visit and other random images. The modified inputs were created by computer-generated distortions, namely:

- Grayscale conversion,
- Brightness increases,
- 50% cropping (25% from top and bottom),
- 90% quality factor JPEG compression,
- Resizing (not respecting dimensions),
- Combinations of the above.

While some of this near-duplicated content is very resemblant to one of the original sequences, some bear little similarities with any input to the naked eye. We fed each one as a new entry to the App. The detection of copies and near-copies matched the performance analysis provided by the authors of [111]. We illustrate this by following the process for three different inputs corresponding to the three different detection cases detailed in Section III-B:

- We give an exact copy of one of the original room visits as input. This file is already stored as is, and our comparison is insensitive to metadata. As expected, a correlation of 1 is found with one of the entries, and the process is rightfully interrupted.
- Then, we feed the App a sequence of images entirely unrelated to any one of the entries. The App does not find any semantic content in the database in the input and indicates the operator as such. If they do, the Smart Contract is called, the database is updated, and an NFT is minted, as Section III explains. The outputs of the App for both instances are shown in Figure 14. In the second image, the operator has prompted the addition of the input in the database, leading to the transactions, of which we show an exploration in the bottom images. After these transactions, if one were to offer the latter file as input once more, the App would answer as in the first image because this input would be recorded as is in the database.
- Finally, to illustrate the ambiguous, near-copy case, we offer a cropped, increased luminosity version of Room 1 to the App. Figure 15 shows the before and after alteration of a sample image of the sequence and the App's response to the input. As dictated by the fingerprinting method's performances, the App computes a 0.78 correlation between the input and one of the entries in the database (the original entry for Room 1). This value (quite comfortably) passes the decided threshold, and the input is detected as a near copy. This result supports the efficiency of the fingerprinting method, which provides robust detection of the semantic content. The App verifies the unicity of the fingerprint in the Smart Contract storage and informs the operator that the entry is a near-copy of a recorded entry. As such, it is not added to the database according to the rules of the use case. If we had decided on a stricter threshold of 0.8, this input would have been considered new and could have been added to the database. However, a museum operator finding both images of Figure 15 in their records would be surprised, as both show the same painting in the same circumstances. This fact illustrates the careful consideration that should be applied to threshold selection.

FIGURE 15. Sample image of the near copy (top) and original counterpart (middle) of Room 1, and the result it generates are being given to the App (bottom).

If the museum wanted to incorporate new rooms into their virtual visit, they would offer a multimedia representation as an input to the workflow, which would confirm its unicity

```
function setup(uint _price, uint _priceCap, uint _cut)
onlyBy(originalOwner) external returns(bool){
```

```
function changePrice(uint _price) onlyBy(owner) external returns(bool){
    require(_price<= priceCap, "The price of this asset is capped.");
    price=_price;
    return(true);
}

function buyToken() payable public {
    require(msg.value == price, "Wrong price sent");
    address payable originalOwnerPay=payable(originalOwner);
    originalOwnerPay.transfer(price*cut/100);
    address payable payableOwner = payable(owner);
    payableOwner.transfer(address(this).balance);
    owner=msg.sender;
}
```

FIGURE 16. Extracts of the TLSC setup for museum tokens. Specifically, the setup function inputs (top) and the `changePrice()` and `buyToken()` functions (bottom) which are called during the lifecycle of the token.

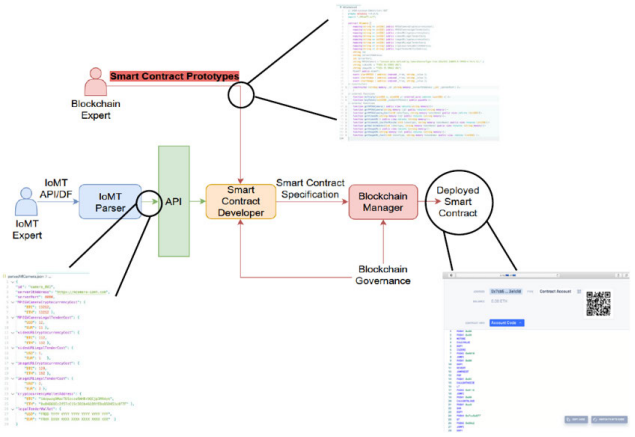


FIGURE 17. Architectural framework for automating the conversion of blockchain IoMT API and data formats into Smart Contracts [38], with focuses on extracts of a Smart Contract Prototype, the output of an IoMT Parser, and an explored Smart Contract, from left to right. The end point of this legacy workflow serves as an input to our architecture, as presented in Figure 18.

before adding it. Upon addition, the App requests the creation of ERC721 NFTs using the Token Prototype. It sends the tokens to the operating wallet of the architecture for extra protection steps to be taken before distribution. Here, we use an RM-TLSC [70] to enforce that a percentage of all subsequent token sales will automatically be sent to the wallet. The same tool can cap the token's price if the museum wants to limit speculation. In Figure 16, we show an extract of this Smart Contract on the Remix online IDE [112], which enables the setting of a museum cut and price cap.

This token can then be sent to a broker or further distributed using any method seen fit by the museum. Otherwise, the token can be kept as proof of the good standing of this original input.

B. IOMT CONTENT TRACING

In this example, we shall illustrate how our architecture and workflow can upgrade and automate an existing solution for monetizing the video content produced by Internet of Media Things (IoMT) cameras. To do so, we use a context where IoMT devices [113], [114], [115] can have their content accessed in a trusted and permissioned fashion by being represented on a blockchain. Specifically, we extend the architecture and use case presented in [38]. For context, Media Things (or *MThings*) – which are defined as *Things* capable of sensing, acquiring, actuating, or processing media content or metadata related with such content – are represented *on-chain* in a systematic fashion using a framework presented in [38] and illustrated hereafter in Figure 17.

The Smart Contracts deployed as such manage the distribution of data collected and produced by the MThings. They do so by creating and monitoring the flow of FTs that serve as access coins through dedicated IoMT interfaces, which they sell via an automatic broker. Potential buyers then interact directly with the broker to acquire these tokens to be redeemed for access to certain content.

Although functional, this software could benefit from the workflow presented in this paper. While this paper's innovation could support the MThing Smart Contracts (produced by the workflow shown in Figure 17) in the control of FTs (*e.g.,* by recording previous buyers), the main advantage it can provide resides in the minting and authentication of an NFT standing for the final access of the content, which can further be used as a Zero-Knowledge Proof (or ZKP) [116]. In [38], the *on-chain* functionalities are limited to the initial purchase of FTs. Although the automatic broker manages a variety of MThings simultaneously and automatically compensates the wallets associated with the MThings, it does not provide support past the purchase of tokens. Further support could track the supply of FTs, potentially restricting undesired behavior. For instance, a given party could potentially hoard these tokens and restrict access to targeted content. By implementing our workflow, the access of this content can be proved and traced authentically, not only through the specification of access tokens that can be limited to given addresses and timeframes but also by creating an NFT timestamping the access to the content.

We approach this complex use case by imagining a single MThing – an MCamera as the sole content provider for illustration purposes. The ideas can be applied to n MThings communicating with the broker, as originally shown in [38]. Once the original architecture and this paper's tools are deployed, the process happens in three steps:

1) A user (through an *off-chain* App) buys access FTs for cryptocurrency or legal tender from an automatized broker. This operator acts as a dam, monitoring the flow of available content and potentially regulating said flow.
2) Using the FTs they bought, users may access/livestream a given amount of feed provided by the MCamera.
3) The access is tokenized and backed, as Section III explains, before being sent to the user as proof of purchase and access to the content. This token's metadata contains information such as MThing characterizations, the buyer, the purchase of FTs, or the time the content was delivered. It can further be used as ZKP.

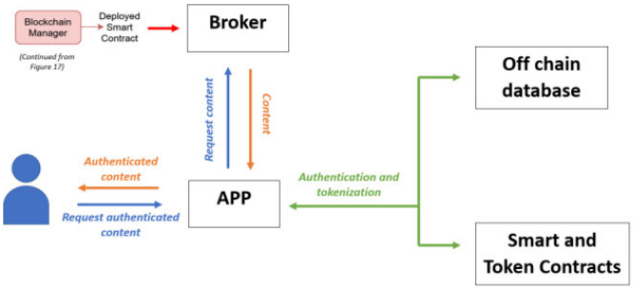


FIGURE 18. Simplified architecture integrating ISO/IEC23093 features. The Broker's input is generated via the workflow shown in Figure 17 and is the entry point of the use case into the novel architecture.

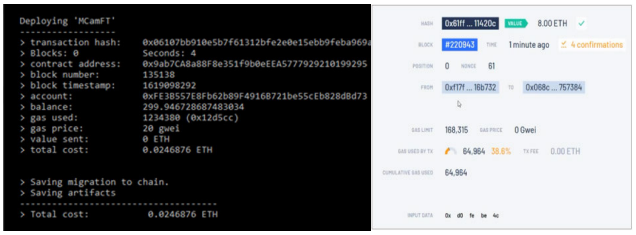


FIGURE 19. Deployment receipt of a Fungible Token Contract used by the IoMT workflow on (left), an explored transaction that occurred automatically during the interaction with the IoMT Broker (right).

Steps 1 and **2** are original to [38], but the *off-chain* Apps can be combined and function as the user's sole contact point with third parties (Broker, Token Contract). A simplified combined architecture is shown in Figure 18.

Specifically, these steps are powered by the architecture and APIs of ISO/IEC23093 series standards (or MPEG-IoMT) [113], [114], [115], which define the data flow and interaction with MThings via blockchains. In [38], the specific interactions between App, blockchain, and MThing in the FT selling and spending process are detailed through a sequence diagram shown in Figure 1 [38].

The deployment of the Token Contract of the MCamera and an example transaction launched under the hood of the IoMT workflow are illustrated in Figure 19. The latter occurs during the exchange shown in the sequence diagram shown in Figure 1 of [38].

Step 3 relies on the architecture brought forth in this paper. Specifically, the identification method of the content should be selected to consider data (the accessed content) and metadata (*e.g.,* timestamps, dates, user). As such, we chose the International Standard Content Code (ISCC) [92], an ISO standardization work item (ISO/AWI 24138, TC 46/DC9/WG18). The ISCC is an open-source, universal, similarity fingerprinting method. ISCC are short and designed for blockchain registration, and they identify any type of content while still being comparable, which could be helpful if multiple MThings were to provide content. ISCC codes comprise four sub-codes: the Meta-Code, Content-Code, Data-Code, and Instance-Code. These codes respectively identify the provider-side metadata, content, data, and user-side metadata. As such, the ISCC generated after two buyers purchase the same content at the same time will remain different, allowing further use and precise identification of the actions of each party, as required by this use case. To do so, codes are compared in subparts, *i.e.,* we calculate the difference between two given codes and the respective differences in each sub-code. While the near duplication of visual content is ensured by combining the Content and Data codes, the Meta and Instance Codes function as checks of the origin of the content on the provider and purchaser side.

Although we mint an ERC721 standard token and consider it a transferable asset that can be protected and sold as its owner wishes, one could keep an eye on the recent advancements in soulbound tokens [117], which are not transferable. This use would reinforce the purpose of the NFT as ZKP but kill other uses for the token. Regardless, a Smart Contract-backed *off-chain* database now records the access of the restricted content and can furthermore sort entries by similarity of data or metadata using unfalsifiable information.

C. SUMMARY

After having detailed our methodology in Sections III and IV, this section illustrates the design philosophy of our solution with regards to set up and further use through an IPR use case (cf. Museum use case) as well as to the integration with high-level solutions (cf. IoMT use case). These use cases bear little resemblance to one another and demonstrate the flexibility of the workflow presented in this paper. In Table 3, we provide a point-by-point comparison of the features supported by our approach with regard to state-of-the-art solutions ([37], [38], [88]), thus showing that the present paper outperforms previous studies. The first three characteristics shown in Table 3 (*i.e.,* near-copy detection, load-balancing architecture, and blockchain-backed storage) deal with functional aspects related to visual content tracking, the two following (*i.e.,* asset tokenization and token distribution) with applicative blockchain concerns, and the last four (*i.e.,* simultaneous standards, simultaneous blockchains, and full lifecycle support, automated workflow) with horizontal features. Green checks signify compliance, red crosses non-compliance, and orange tildes partial compliance.

In the following section, we go further in analyzing the solution presented in this paper, discussing its advantages and drawbacks and its applicative scope and performance ceilings.

VI. ANALYSIS

In Section III, we introduced the fundamental ideas of our architecture and its general workflow. Section IV illustrated the general technological bricks we make use of, and Section V implemented the architecture in the context of real-world scenarios. This section will provide a critical retrospective view of our solution.

TABLE 3. Comparative analysis of our solution with regards to state-of-the-art solutions.

	State-of-the-art solutions			Advanced solution
	[37]	[38]	[88]	
Near-copy detection	✓	✗	✗	✓
Load-balancing architecture	✓	✗	✓	✓
Blockchain-backed storage	✓	✗	✓	✓
Asset Tokenization	≈	≈	✗	✓
Token distribution	✗	✗	✗	✓
Simultaneous standards	≈	≈	✗	✓
Simultaneous blockchains	✗	✗	✗	✓
Automated workflow	✗	✗	✗	✓
Full lifecycle support	✗	✗	✗	✓

The ideas presented in this paper combine various visual content and blockchain technologies to provide automated, end-to-end lifecycle management of authenticated assets on blockchain environments. As such, our architecture and workflow design results from the strengths and limitations of each of its technologies and their combination. Hereafter, we analyze the main benefits and shortcomings of our work through the lens of four aspects, specifically: the use case reliance, the flexibility of the architecture through the association of technologies, the key features of the architecture, and its overarching technical challenges.

① Use case reliance: This architecture's most significant advantage and drawback is its intimate symbiosis with the specifics of the applicative scenario. Throughout Sections III and IV, we reiterated that the optimization of most technological bricks was heavily reliant on use cases. Although specified and prototyped, the architecture and workflow are only completely defined within a context they are tailor-made to suit. As such, it is natural that the method be performant within these set bounds at the expense of being defined by them.

Although the most recurrent characteristics could be standardized into a cookie-cutter adapted to most use cases, further work should instead focus on applying the customization processes we showed in Section V to contingent workflows that would benefit from the creation and authentication of assets.

② Technology association and flexibility: An essential aspect of our architecture is the performance preservation of each part. The architecture does not impose any extra constraints on the methods it accommodates. This allows, for instance, the identification method to be selected according to its format and robustness to various kinds of visual content attacks, the inclusion of metadata, or any other feature. The same can be said with most technologies used in the workflow: not only are the database solution, App programming language, blockchain infrastructure, and token standards interchangeable, but the very input formats can also be modified to suit specific requirements (*e.g.,* to an audio-oriented use case). Current-day *on-chain* environments lack precise semantic content identification that similarity-preserving fingerprints can bring thanks to this workflow. The association of both provides a robust framework for identifying semantic content, which can be put at the service of a wide array of scenarios thanks to the flexibility of the workflow. Nevertheless, the advanced workflow is flexible in its expansion, enabling multifaceted management with simple additions, and our solution is conceived, designed, and implemented to let the near duplicated content detection be substituted without breaking the workflow. If a use case did not need to worry itself with near-duplication and semantic content concerns, a hashing method could substitute fingerprinting without affecting the nature of the processing chain. Consequently, our architecture and workflow can seamlessly integrate a broad spectrum of standard visual identification technologies, which we illustrated using two very different algorithms in our implementation section. Further media technologies for speech, audio, or 3D data could also be employed similarly or in combination with visual identification, *e.g.,* using MPEG Compact Descriptors for Video Recognition.

We already showed that our findings could be valuable add-ons to existing solutions, extending their functionalities and automating the process, as in Section V-B. Hence, future applicative work could enable this solution in new environments such as the metaverse. From a methodological point of view, future work could explore technologies capable of supporting further features in each architecture brick and use emerging mixed token standards to enable the multifaceted reliance on a single Token Contract.

③ Architectural features: When considering the workflow supported by our architecture, a key element is the complete lifecycle support of assets. The workflow begins with multiple steps before creating a blockchain asset and ends with distributing the new asset. This wide range of operations notably allows the tailoring of the asset and integration of complementary state-of-the-art solutions related, for instance, to royalties or brokering.

We can also note the simplicity of using this solution. Once a qualified expert has set the architecture up, operators without knowledge of databases and blockchain development can use it. The App serves as the sole interaction point for the operator, primarily through simple prompts. The workflow interrupts itself in case of database tampering, preventing the operator from knowingly or unknowingly validating malicious behavior. This feature is a significant benefit for solution's acceptance, which remains a significant issue for any new blockchain technology. Additionally, this

solution's end users are institutions possessing diverse media content with IPR to be applied to web3 environments, not individuals with minimal resources to allocate.

The perennity and privacy of user data are also ensured by the fact that only a dedicated wallet interacts with the blockchain at the command of the App. Although our use cases did not require the *on-chain* storage of sensitive data, extensions that would require such a feature could rely on the fact that said data only appears through the hashes of its visual fingerprints and, hence, is not exposed when publicly accessible. Further, as dictated by best practices and blockchain *de facto* standards, the Smart Contract used in the architecture is very simple and lightweight, ensuring its purpose is fulfilled while maintaining its blockchain-inherited security.

Furthermore, our solution showcases various interoperability features of the software and its components, although we do not deal with an Interoperability Mechanism (IM) as defined by [118]. We ensured the interchangeability and systematic integration of diverse state-of-the-art methodological bricks from *on* and *off-chain* environments, as mentioned in ② . The workflow uses the fundamental features of each technology it accommodates while benefiting from more complex features. This is true for the *off-chain* data storage, blockchain environments and standards, and the App in and of itself. For instance, all modern, application-oriented blockchains seamlessly interacting with an *off-chain* App can be used indiscriminately. Moreover, the workflow output carries traces of these solutions and is used in standard ways throughout the environment. This feature is the case with a broad spectrum of potential use cases related to various fields, two examples of which are given in Section V. Moreover, the App can manage Smart Contracts throughout multiple blockchain environments simultaneously. As such, this workflow promotes interoperability at the level of a blockchain but also between blockchains.

Scalability is always of significant concern when dealing with blockchains. Although our methodology does not act at the level of the infrastructure or protocol, it is designed to enable seamless expandability. First, all the blockchain data is stored in web3-friendly formats (*e.g.,* ISCC fingerprints). Fingerprints not supporting this feature are systematically adjusted to minimize resource usage via hashing functions. Second, features that typically linearly scale with the number of entries in the database, namely the greenlight process, which occurs at every call of the Smart Contract, use value mapping and call-only operations that do not write information on the blockchain (*i.e.,* do not constitute transactions). The transactions only occur once an entry is appended, and results in changing a single Boolean value and in the optional subsequent token minting. In its *off-chain* components, content is only treated a single time in its potential admission, which puts the limiting scalability factor in the fingerprint treatment capacity of the database. Although database performance is not in our area of expertise, we can note that not only are our required entries light (*e.g.*, <1MB for our most advanced video fingerprint), but they can benefit from further compression, which has been an active research field for over 30 years. Lastly, the Token Contract is the only element requiring any extra instantiation in expanding the architecture to support multiple databases and formats simultaneously (*cf.* Section IV). As such, our approach shows its benefits in scalability, which trickle down to its resource consumption and cost, as presented in ④.

Future work could formalize IMs provided by this workflow but also use Inter-Blockchain Communication (IBC) [119]-oriented IMs being explored, specifically those enabling token exchanges across infrastructures. Potential additions should remain mindful of retaining the straightforwardness of the solution.

④ Technical challenges: The central presence of blockchain processing in this architecture brings native blockchain security mechanisms into the workflow. In our case, a Smart Contract acts as a zero-trust third party that verifies the integrity of the database used to authenticate assets. As such, the workflow is as secure as the link between the App and blockchain, which lies in sound daily security practices and careful private key management. However, the presence of a blockchain also naturally brings other points of the question, notably in terms of computational efficiency and energy consumption.

Regarding computational efficiency, the bulk of the computational power required for our workflow is concentrated in the initial setup. After the setup and for routine use, the Smart Contract is only invoked at two specific moments, limiting the resource consumption of this process. The first use of the Smart Contract is the greenlight function and does not constitute a transaction as it does not write any information on the blockchain. This call does not cost gas and is not limited by slow block rates. The second is the addition of a new entry in the database. This step is the initial setup brought to the scale of a single entry. A single line of code in the Smart Contract setting a Boolean variable to True is sufficient for this task. As was our aim, this design leaves the heavy lifting to the *off-chain* technological bricks. Thus, we avoid the brunt of the central issue of *on-chain* development: lengthy processing times and resource intensity.

Resource consumption on the blockchain is measured via gas, whose costs are relative to specific blockchains and network congestion. Moreover, the gas price on the largest blockchains is also subject to financial market fluctuations. The workflow presented in this paper minimizes these dependencies by only calling its blockchain elements (Smart and Token Contracts) in the event of successfully processing a new verified entry, which limits post-deployment gas spending to a minimum. Regarding the initial deployment, no strategies other than code optimization can curb the cost of instantiating software on blockchains. For instance, the deployment of the Smart and Token Contracts of the Museum IPR use case we presented in Section V respectively used 15.24% and 61.45% of the default 4.5M gas limit of our private EEA blockchain (*i.e.,* approx. 685,000 and 2.7M). These values are made possible by the load-balancing

features of the architecture, without which most fingerprints, even if they were supported, would be prohibitively expensive to be stored on a blockchain. To put these numbers in perspective, [120] estimates 640k gas being necessary to store a kilobyte of data on Ethereum's mainnet. As such, a single fingerprint computed with [111], as we use in Section V-A, would cost 22.4M gas to be stored, which is higher than Ethereum's total block gas target of 15M [121]. For comparison, a simple transaction of ETH from one account to another account costs about 21,000 gas, while an ETH transaction to a Smart Contract costs 68,000 in the same circumstances. As such, the processing times and costs brought forth are well within blockchain standards and participate in the performance preservation discussed in ②.

Of course, these gas and performance costs relate to real-world spending and energy consumption proportionately to the mindfulness of the blockchain's consensus algorithm and energetical mindfulness, as well as absolute gas prices in legal tender. We decided to employ the Ethereum and Tezos blockchains, which both use variations on the Proof of Stake consensus algorithm. This method has now widely succeeded energy-hungry Proof of Work in the applicative blockchain realm in a general effort to build an environmentally conscious web3 ecosystem. The consensus shift from Ethereum 1.0 to Ethereum 2.0 lowered its energy consumption by a staggering 99.95%, now making the blockchain consume ten times less energy than AirBnB at approximately 0.0026 TWh per year. Tezos is also one of the most energy-responsible blockchains, consuming even less than the newly efficient Ethereum 2.0. Moreover, this workflow does not aim to create assets that would not have been created without it but to support said assets and assert their link with their real-world or web2 counterparts. Gas is what is paid in exchange for the authentication provided by this workflow.

Future work could apply the efforts presented in this paper in the light of so-called green blockchains [122], [123] and energy-conscious blockchain discoveries to ensure the benefit provided by our workflow can come at a minimal cost.

VII. CONCLUSION

Web3 paradigms brought and continue to bring new high-stake threats to multimedia content in terms of misuse (e.g., copying, IPR infringement) but also carry the opportunity of more robust protection schemes. In this paper, we bring forward a flexible architecture that enables the backing of semantic information from given datasets in a blockchain-supported fashion and its subsequent usage through automated workflows. Specifically, this architecture makes the most out of the mutually beneficial connection between off-chain and on-chain technologies to provide advanced multimedia identification and data integrity. Data authenticated as such is then made into blockchain assets that bear the trace of the process and can be distributed or serve as ZKPs.

This solution can rely on various web2 and web3 technologies and be implemented standalone or non-invasively within already established workflows. It avoids the biggest blockchain pitfalls by keeping slow and costly operations at a minimum yet ensuring the best out of the robustness provided by the technology. These features widen its spectrum of potential uses, as we illustrated throughout this paper.

Further efforts should not only integrate this solution into new use cases to face new challenges but also keep it up to date with new web3 concepts such as emerging mixed token standards, IBC, and green blockchains.

APPENDIX—ABBREVIATIONS

Abbreviations used more than once throughout this paper are summarized in Table 4.

TABLE 4. Abbreviations used in this paper.

API	Application Programming Interface
EEA	Ethereum Enterprise Alliance
EIP	Ethereum Improvement Proposals
ERC	Ethereum Request for Comments
FT	Fungible Token
IBC	Inter Blockchain Communication
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IM	Interoperability Mechanism
IoMT	Internet of Media Things
IP	Intellectual Property
IPR	Intellectual Property Rights
ISCC	International Standard Content Code
ISO	International Organization for Standardization
MPEG	Moving Picture Experts Group
MThing	Media Thing
NFT	Non-Fungible Token
RM-TLSC	Royalty Management Token Level Smart Contract
ZKP	Zero-Knowledge Proof

REFERENCES

[1] H. Lippell, "Big data in the media and entertainment sectors," in *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*. Cham, Switzerland: Springer, 2016, pp. 245–259, doi: 10.1007/978-3-319-21569-3_14.

[2] L. Cao, G.-J. Qi, S.-F. Tsai, M.-H. Tsai, A. Pozo, X. Zhang, and S. Lim, "Multimedia information networks in social media," in *Social Network Data Analytics*. Boston, MA, USA: Springer, 2011, pp. 413–445, doi: 10.1007/978-1-4419-8462-3_15.

[3] S.-C. Chen, "Multimedia for autonomous driving," *IEEE Multimedia*, vol. 26, no. 3, pp. 5–8, Jul. 2019, doi: 10.1109/MMUL.2019.2935397.

[4] W. Zhu, X. Wang, and W. Gao, "Multimedia intelligence: When multimedia meets artificial intelligence," *IEEE Trans. Multimedia*, vol. 22, no. 7, pp. 1823–1835, Jul. 2020, doi: 10.1109/TMM.2020.2969791.

[5] Grand View Research. *Digital Content Creation Market Size, Share & Trends Analysis*. Accessed: Jul. 2023. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/digital-content-creation-market-report

[6] H. Abbas and R. Di Pietro, "Sanitization of visual multimedia content: A survey of techniques, attacks, and future directions," 2022, *arXiv:2207.02051*.

[7] A. Koschmann and Y. Qian, "Latent estimation of piracy quality and its effect on revenues and distribution: The case of motion pictures," White Paper 27649, Nat. Bureau Econ. Res., Cambridge, MA, USA, 2020, doi: 10.3386/w27649.

[8] L. Ma, A. L. Montgomery, P. V. Singh, and M. D. Smith, "An empirical analysis of the impact of pre-release movie piracy on box office revenue," *Inf. Syst. Res.*, vol. 25, no. 3, pp. 590–603, Sep. 2014, doi: 10.1287/isre.2014.0530.

[9] L. Cao, "Decentralized AI: Edge intelligence and smart blockchain, metaverse, Web3, and DeSci," *IEEE Intell. Syst.*, vol. 37, no. 3, pp. 6–19, May 2022, doi: 10.1109/MIS.2022.3181504.

[10] W. Ding, J. Hou, J. Li, C. Guo, J. Qin, R. Kozma, and F.-Y. Wang, "DeSci based on Web3 and DAO: A comprehensive overview and reference model," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 5, pp. 1563–1573, Oct. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9906878

[11] Z. Liu, Y. Xiang, J. Shi, P. Gao, H. Wang, X. Xiao, B. Wen, Q. Li, and Y.-C. Hu, "Make Web3.0 connected," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 2965–2981, Sep. 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9428608

[12] A. Qureshi and D. M. Jiménez, "Blockchain-based multimedia content protection: Review and open challenges," *Appl. Sci.*, vol. 11, no. 1, p. 1, Dec. 2020, doi: 10.3390/app11010001.

[13] S.-C. Chen, "Multimedia research toward the metaverse," *IEEE Multimedia*, vol. 29, no. 1, pp. 125–127, Jan./Mar. 2022, doi: 10.1109/MMUL.2022.3156185.

[14] H. Dong and J. S. A. Lee, "The metaverse from a multimedia communications perspective," *IEEE Multimedia*, vol. 29, no. 4, pp. 123–127, Oct./Dec. 2022, doi: 10.1109/MMUL.2022.3217627.

[15] X. Xu, G. Zou, L. Chen, and T. Zhou, "Metaverse space ecological scene design based on multimedia digital technology," *Mobile Inf. Syst.*, vol. 2022, Jul. 2022, Art. no. 7539240, doi: 10.1155/2022/7539240.

[16] D. Buhalis, M. S. Lin, and D. Leung, "Metaverse as a driver for customer experience and value co-creation: Implications for hospitality and tourism management and marketing," *Int. J. Contemp. Hospitality Manage.*, vol. 35, no. 2, pp. 701–716, Jan. 2023, doi: 10.1108/IJCHM-05-2022-0631.

[17] H. Duan, J. Li, S. Fan, Z. Lin, X. Wu, and W. Cai, "Metaverse for social good: A university campus prototype," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 153–161, doi: 10.1145/3474085.3479238.

[18] A. Nadeem and M. Y. Javed, "A performance comparison of data encryption algorithms," in *Proc. Int. Conf. Inf. Commun. Technol.*, Karachi, Pakistan, Aug. 2005, pp. 84–89, doi: 10.1109/ICICT.2005.1598556.

[19] S. R. Subramanya and B. K. Yi, "Digital signatures," *IEEE Potentials*, vol. 25, no. 2, pp. 5–8, Mar./Apr. 2006, doi: 10.1109/MP.2006.1649003.

[20] C. I. Podilchuk and E. J. Delp, "Digital watermarking: Algorithms and applications," *IEEE Signal Process. Mag.*, vol. 18, no. 4, pp. 33–46, Jul. 2001, doi: 10.1109/79.939835.

[21] B. Rogosky and R. Goldstone, "Adaptation of perceptual and semantic features," in *Functional Features in Language and Space: Insights from Perception, Categorization, and Development*, L. Carlson and E. van der Zee, Eds. Oxford, U.K.: Oxford Academic, Jan. 2010, doi: 10.1093/acprof:oso/9780199264339.003.0017.

[22] J. Lu, "Video fingerprinting for copy identification: From research to industry applications," *Proc. SPIE*, vol. 7254, Feb. 2009, Art. no. 725402, doi: 10.1117/12.805709.

[23] C. Antal, T. Cioara, I. Anghel, M. Antal, and I. Salomie, "Distributed ledger technology review and decentralized applications development guidelines," *Future Internet*, vol. 13, no. 3, p. 62, Feb. 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/3/62

[24] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," Natl. Inst. Stand. Technol., Gaithersburg, MD, USA, Tech. Rep. 800-207, Aug. 2020, pp. 1–59, doi: 10.6028/NIST.SP.800-207.

[25] C. Buck, C. Olenberger, A. Schweizer, F. Völter, and T. Eymann, "Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust," *Comput. Secur.*, vol. 110, Nov. 2021, Art. no. 102436, doi: 10.1016/j.cose.2021.102436.

[26] T. M. Navamani, "A review on cryptocurrencies security," *J. Appl. Secur. Res.*, vol. 18, no. 1, pp. 49–69, Jan. 2023, doi: 10.1080/19361610.2021.1933322.

[27] S. Biswas, K. Sharif, F. Li, I. Alam, and S. P. Mohanty, "DAAC: Digital asset access control in a unified blockchain based e-health system," *IEEE Trans. Big Data*, vol. 8, no. 5, pp. 1273–1287, Oct. 2022, doi: 10.1109/TBDATA.2020.3037914.

[28] V. T. Truong, L. Le, and D. Niyato, "Blockchain meets metaverse and digital asset management: A comprehensive survey," *IEEE Access*, vol. 11, pp. 26258–26288, 2023.

[29] H. R. Hasan and K. Salah, "Proof of delivery of digital assets using blockchain and smart contracts," *IEEE Access*, vol. 6, pp. 65439–65448, 2018, doi: 10.1109/ACCESS.2018.2876971.

[30] T. Schrepel, "The complex relationship between Web2 giants and Web3 projects," Amsterdam Law & Technol. Inst. Working Paper 1-2023, VU Univ. Amsterdam Legal Stud., Amsterdam, The Netherlands, Jan. 10, 2023, doi: 10.2139/ssrn.4284597.

[31] B. Notheisen, J. B. Cholewa, and A. P. Shanmugam, "Trading real-world assets on blockchain," *Bus. Inf. Syst. Eng.*, vol. 59, pp. 425–440, Oct. 2017, doi: 10.1007/s12599-017-0499-8.

[32] X. Min, L. Kong, Q. Li, Y. Liu, B. Zhang, Y. Zhao, Z. Xiao, and B. Guo, "Blockchain-native mechanism supporting the circulation of complex physical assets," *Comput. Netw.*, vol. 202, Jan. 2022, Art. no. 108588, doi: 10.1016/j.comnet.2021.108588.

[33] Y. Chen and C. Bellavitis, "Blockchain disruption and decentralized finance: The rise of decentralized business models," *J. Bus. Venturing Insights*, vol. 13, Jun. 2020, Art. no. e00151, doi: 10.1016/j.jbvi.2019.e00151.

[34] R. Lombardi, C. de Villiers, N. Moscariello, and M. Pizzo, "The disruption of blockchain in auditing—A systematic literature review and an agenda for future research," *Accounting, Auditing Accountability J.*, vol. 35, no. 7, pp. 1534–1565, Aug. 2022, doi: 10.1108/AAAJ-10-2020-4992.

[35] V. Rambhia, V. Mehta, R. Mehta, R. Shah, and D. Patel, "Intellectual property rights management using blockchain," in *Information and Communication Technology for Competitive Strategies (ICTCS 2020)* (Lecture Notes in Networks and Systems), vol. 190, M. S. Kaiser, J. Xie, and V. S. Rathore, Eds. Singapore: Springer, 2022, doi: 10.1007/978-981-16-0882-7_47.

[36] M. Allouche, T. Frikha, M. Mitrea, G. Memmi, and F. Chaabane, "Lightweight blockchain processing. Case study: Scanned document tracking on Tezos blockchain," *Appl. Sci.*, vol. 11, no. 15, p. 7169, Aug. 2021, doi: 10.3390/app11157169.

[37] A. Moreaux and M. Mitrea, "Visual content verification in blockchain environments," *Blockchain Cryptocurrency*, vol. 1, no. 1, pp. 44–55, Sep. 2023.

[38] M. Allouche, M. Mitrea, A. Moreaux, and S.-K. Kim, "Automatic smart contract generation for Internet of Media Things," *ICT Exp.*, vol. 7, no. 3, pp. 274–277, Sep. 2021, doi: 10.1016/j.icte.2021.08.009.

[39] R. Sobti and G. Ganesan, "Cryptographic hash functions: A review," *Int. J. Comput. Sci. Issues*, vol. 9, no. 2, pp. 461–479, 2012.

[40] J. Liu, Z. Huang, H. Cai, H.T. Shen, C.W. Ngo, and W. Wang, "Near duplicate video retrieval: Current research and future trends," *ACM Comput. Surv.*, vol. 45, no. 4, pp. 1–23, 2013, doi: 10.1145/2501654.2501658.

[41] J. Oostveen, T.Kalker, and J. Haitsma, "Feature extraction and a database strategy for video fingerprinting," in *Recent Advances in Visual Information Systems* (Lecture Notes in Computer Science), vol. 2314, S. K. Chang, Z. Chen, and S. Y. Lee, Eds. Berlin, Germany: Springer, 2002, doi: 10.1007/3-540-45925-1_11.

[42] A. Saracoglu, E. Esen, T. K. Ates, B. O. Acar, U. Zubari, E. C. Ozan, E. Ozalp, A. A. Alatan, and T. Ciloglu, "Content based copy detection with coarse audio-visual fingerprints," in *Proc. 7th Int. Workshop Content-Based Multimedia Indexing*, Chania, Greece, Jun. 2009, pp. 213–218, doi: 10.1109/CBMI.2009.12.

[43] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, "DeepMarks: A secure fingerprinting framework for digital rights management of deep learning models," in *Proc. Int. Conf. Multimedia Retr. (ICMR)*. New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 105–113, doi: 10.1145/3323873.3325042.

[44] G. Conti and K. Abdullah, "Passive visual fingerprinting of network attack tools," in *Proc. ACM Workshop Visualizat. Data Mining Comput. Secur. (VizSEC/DMSEC)*. New York, NY, USA: Association for Computing Machinery, Oct. 2004, pp. 45–54, doi: 10.1145/1029208.1029216.

[45] M. Allouche and M. Mitrea, "Video fingerprinting: Past, present, and future," *Frontiers Signal Process.*, vol. 2, Sep. 2022, Art. no. 984169, doi: 10.3389/frsip.2022.984169.

[46] Y.-G. Jiang and J. Wang, "Partial copy detection in videos: A benchmark and an evaluation of popular methods," *IEEE Trans. Big Data*, vol. 2, no. 1, pp. 32–42, Mar. 2016, doi: 10.1109/TBDATA.2016.2530714.

[47] M. Wu, W. Trappe, Z. J. Wang, and K. J. R. Liu, "Collusion-resistant fingerprinting for multimedia," in *IEEE Signal Process. Mag.*, vol. 21, no. 2, pp. 15–27, Mar. 2004, doi: 10.1109/MSP.2004.1276103.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[49] J. Jin, X. Zhang, X. Fu, H. Zhang, W. Lin, J. Lou, and Y. Zhao, "Just noticeable difference for deep machine vision," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 6, pp. 3452–3461, Jun. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9540665

[50] G. Kordopatis-Zilos, S. Papadopoulos, I. Patras, and Y. Kompatsiaris, "Near-duplicate video retrieval with deep metric learning," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Venice, Italy, Oct. 2017, pp. 347–356, doi: 10.1109/ICCVW.2017.49.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[52] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Bitcoin, White paper, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[53] V. Buterin, "A next-generation smart contract and decentralized application platform," Ethereum, White Paper, pp. 2–1, 2014, vol. 3, no. 37. [Online]. Available: https://ethereum.org/en/whitepaper/

[54] Gartner Peer Insights. *Blockchain Platforms Reviews 2021*. Accessed: Jun. 2023. [Online]. Available: https://www.gartner.com/reviews/market/blockchain-platforms

[55] CoinMarketCap. *Coin Market Cap. For All Coins 2021*. Accessed: May 2023. [Online]. Available: https://coinmarketcap.com/coins/views/all/

[56] N. Szabo, "Formalizing and securing relationships on public networks," *1st Monday*, vol. 2, no. 9, pp. 1–25, Sep. 1997, doi: 10.5210/fm.v2i9.548.

[57] S. Rouhani and R. Deters, "Security, performance, and applications of smart contracts: A systematic survey," *IEEE Access*, vol. 7, pp. 50759–50779, 2019, doi: 10.1109/ACCESS.2019.2911031.

[58] S. M. H. Bamakan, A. Motavali, and A. B. Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria," *Expert Syst. Appl.*, vol. 154, Sep. 2020, Art. no. 113385, doi: 10.1016/j.eswa.2020.113385.

[59] W. Zou, D. Lo, P. S. Kochhar, X. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Trans. Softw. Eng.*, vol. 47, no. 10, pp. 2084–2106, Oct. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8847638

[60] W. Liang, Y. Fan, K. -C. Li, D. Zhang, and J. -L. Gaudiot, "Secure data storage and recovery in industrial blockchain network environments," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6543–6552, Oct. 2020, doi: 10.1109/TII.2020.2966069.

[61] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale Internet of Things data storage and protection," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 762–771, Sep. 2019, doi: 10.1109/TSC.2018.2853167.

[62] IEEE Xplore. *IEEE Blockchains Standards*. Accessed: Jun. 2023. [Online]. Available: https://innovate.ieee.org/ieee-blockchain-standards-collection/

[63] ISO. *ISO/TC 307 Blockchain and Distributed Ledger Technologies*. Accessed: Jun. 2023. [Online]. Available: https://www.iso.org/committee/6266604.html

[64] *International Association for Trusted Blockchain Applications*. Accessed: Jun. 2023. [Online]. Available: https://inatba.org/

[65] *Enterprise Ethereum Alliance*. Accessed: May 2023. [Online]. Available: https://entethalliance.org/

[66] F. Vogelsteller and V. Buterin. (Nov. 2015). *ERC-20: Token Standard*. Ethereum Improvement Proposals No. 20. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20

[67] W. Entriken, D. Shirley, J. Evans, and N. Sachs. (Jan. 2018). *EIP-721: Non-Fungible Token Standard*. Ethereum Improvement Proposals No. 721. [Online]. Available: https://eips.ethereum.org/EIPS/eip-721

[68] W. Radomski, A. Cooke, P. Castonguay, J. Therien, E. Binet, and R. Sandford. (Jun. 2018). *ERC-1155: Multi Token Standard*. Ethereum Improvement Proposals No. 1155. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1155

[69] Consensys. *An Analysis of Ethereum's Decentralized Finance Ecosystem in Q2 2021*. DeFi Report Q2 2021. Accessed: Mar. 2023. [Online]. Available: https://consensys.net/reports/defi-report-q2-2021

[70] A. C. Moreaux and M. P. Mitrea, "Royalty-friendly digital asset exchanges on blockchains," *IEEE Access*, vol. 11, pp. 56235–56247, 2023, doi: 10.1109/ACCESS.2023.3283153.

[71] N. Kshetri, "Scams, frauds, and crimes in the nonfungible token market," *Computer*, vol. 55, no. 4, pp. 60–64, Apr. 2022, doi: 10.1109/MC.2022.3144763.

[72] D. Das, P. Bose, N. Ruaro, C. Kruegel, and G. Vigna, "Understanding security issues in the NFT ecosystem," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: Association for Computing Machinery, Nov. 2022, pp. 667–681, doi: 10.1145/3548606.3559342.

[73] S. Saha Roy, D. Das, P. Bose, C. Kruegel, G. Vigna, and S. Nilizadeh, "Demystifying NFT promotion and phishing scams," 2023, *arXiv:2301.09806*.

[74] *Cryptocurrencies: Last Week Tonight with John Oliver (HBO)*. YouTube. Accessed: Feb. 2023. [Online]. Available: https://www.youtube.com/watch?v=g6iDZspbRMg

[75] J. Sor. *Who is Cofeezilla, the Crypto Detective Who Says He Got Sam Bankman-Fried to Admit to Fraud*. Business Insider. Accessed: Jun. 2023. [Online]. Available: https://markets.businessinsider.com/news/currencies/coffeezilla-sam-bankman-fried-ftx-bankruptcy-crypto-scam-detective-2023-1

[76] R. Monroe. *Coffeezilla, the YouTuber Exposing Crypto Scams*. The New Yorker. Accessed: Jun. 2023. [Online]. Available: https://www.newyorker.com/news/letter-from-the-southwest/coffeezilla-the-youtuber-exposing-crypto-scams

[77] The Block. *Ethereum NFT Marketplace Monthly Volume*. Accessed: May 2023. [Online]. Available: https://www.theblock.co/data/nft-non-fungible-tokens/marketplaces/nft-marketplace-monthly-volume

[78] M. K. Manoylov. *OpenSea Reveals that Over 80% of Its Free NFT Mints were Plagiarized, Spam or Fake*. The Block. Accessed: Feb. 2023. [Online]. Available: https://www.theblock.co/linked/132511/opensea-reveals-that-over-80-of-its-free-nft-mints-were-plagiarized-spam-or-fake

[79] (2020). *The Tokenisation of Assets and Potential Implications for Financial Markets*. OECD Blockchain Policy Series. [Online]. Available: https://www.oecd.org/finance/The-Tokenisation-of-Assets-and-Potential-Implications-for-Financial-Markets.htm

[80] R. Heines, C. Dick, C. Pohle, and R. Jung, "The tokenization of everything: Towards a framework for understanding the potentials of tokenized assets," in *Proc. PACIS*, Jul. 2021, pp. 1–15.

[81] S. M. H. Bamakan, N. Nezhadsistani, O. Bodaghi, and Q. Qu, "Patents and intellectual property assets as non-fungible tokens; key technologies and challenges," *Sci. Rep.*, vol. 12, p. 2178, Feb. 2022, doi: 10.1038/s41598-022-05920-6.

[82] A. Thorn, M. Marcantonio, and G. Parker. *A Survey of NFT Licenses: Facts & Fictions*. Galaxy. Accessed: May 2023. [Online]. Available: https://www.galaxy.com/research/insights/a-survey-of-nft-licenses-facts-and-fictions/

[83] Creative Commons. *About CC Licenses*. Accessed: Jul. 2023. [Online]. Available: https://creativecommons.org/about/cclicenses/

[84] S. Qadir and G. Parker. *NFT Royalties: The $1.8bn Question*. Galaxy. Accessed: Feb. 2023. [Online]. Available: https://www.galaxy.com/research/insights/nft-royalties/

[85] Z. Burks, J. Morgan, B. Malone, and J. Seibel. (Sep. 2020). *EIP-2981: NFT Royalty Standard*. Ethereum Improvement Proposals No. 2981. [Online]. Available: https://eips.ethereum.org/EIPS/eip-2981

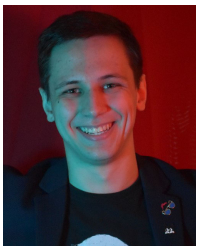
[86] R. Li, "Fingerprint-related chaotic image encryption scheme based on blockchain framework," *Multimedia Tools Appl.*, vol. 80, no. 20, pp. 30583–30603, Aug. 2021, doi: 10.1007/s11042-020-08802-z.

[87] F. Frattolillo, "A watermarking protocol based on blockchain," *Appl. Sci.*, vol. 10, no. 21, p. 7746, Nov. 2020, doi: 10.3390/app10217746.

[88] M. Allouche, M. Ljubojevic, and M. Mitrea, "Visual document tracking and blockchain technologies in mobile world," in *Proc. Electron. Imag. Int. Conf. Imag. Multimedia Anal. Web Mobile World*, Jan. 2021, pp. 279-1–279-6, doi: 10.2352/ISSN.2470-1173.2021.8.IMAWM-279.

[89] A. Moreauxa and M. Mitrea, "Blockchain assisted near-duplicated content detection," in *Proc. Blockchain Cryptocurrency Congr. (B2C)*, 2022, pp. 98–139.

[90] L. Tseng, X. Yao, S. Otoum, M. Aloqaily, and Y. Jararweh, "Blockchain-based database in an IoT environment: Challenges, opportunities, and analysis," *Cluster Comput.*, vol. 23, pp. 2151–2165, Jul. 2020, doi: 10.1007/s10586-020-03138-7.
[91] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2017, pp. 468–477, doi: 10.1109/CCGRID.2017.8.
[92] *International Standard Content Code Foundation Portal*. Accessed: Mar. 2023. [Online]. Available: https://iscc.foundation/iscc/
[93] J. Noble, A. K. P. Taivalsaari, and I. Moore, *Prototype-Based Programming: Concepts, Languages and Applications*. Singapore: Springer, 1999.
[94] *ISO/IEC 21000-23 Smart Contracts for Media*. Accessed: Jan. 2023. [Online]. Available: https://scm.linkeddata.es/
[95] ISO. *ISO/IEC 21000-23:2022(EN) Information Technology—Multimedia Framework (MPEG-21)—Part 23: Smart Contracts for Media*. Accessed: May 2023. [Online]. Available: https://www.iso.org/obp/ui/fr/#iso:std:iso-iec:21000:-23:ed-1:v1:en
[96] GitHub. *Royalty Token Repository*. Accessed: Aug. 2023. [Online]. Available: https://github.com/a-moreaux/RoyaltyToken
[97] SmartPy. *SmartPy RM-TLSC Demo Contract*. Accessed: Aug. 2023. [Online]. Available: https://legacy.smartpy.io/ide?cid=QmSyFVp5m7RDj9SyLyspEsLvfaNcGL8TYeHvReNFLTzoGt&k=5167c9ceb09f5ff9ae5c
[98] *SmartPy Tezos IDE*. Accessed: Jun. 2023. [Online]. Available: https://smartpy.io/
[99] SmartPy. *Legacy SmartPy Tezos IDE*. Accessed: Jun. 2023. [Online]. Available: https://legacy.smartpy.io/
[100] SmartPy. *Meta-Programming*. Accessed: Feb. 2023. [Online]. Available: https://legacy.smartpy.io/docs/introduction/meta_programming#meta-programming
[101] SmartPy. *SmartPy Smart Contract Database Backing*. Accessed: Apr. 2023. [Online]. Available: https://legacy.smartpy.io/ide?cid=QmZa8H7PPHDXtmEk5umZqFsSS5R4wBmLcTr7WxDSvNqHwk&k=24deab4dc14ccfda82ca
[102] TzKT. *Tezos Blockchain Explorer*. Accessed: Jun. 2023. [Online]. Available: https://tzkt.io/
[103] Tezos Deverloper Portal. *FA2 A Unified Token Contract Interface*. Accessed: Sep. 2023. [Online]. Available: https://tezos.b9lab.com/fa2
[104] OpenZeppelin Docs. *ERC721*. Accessed: Mar. 2023. [Online]. Available: https://docs.openzeppelin.com/contracts/2.x/api/token/erc721
[105] OpenZeppelin Docs. *ERC1155*. Accessed: Mar. 2023. [Online]. Available: https://docs.openzeppelin.com/contracts/3.x/erc1155
[106] *OpenZeppelin Token Wizard*. Accessed: Apr. 2023. [Online]. Available: https://wizard.openzeppelin.com/
[107] Web3Py. *Web3Py Documentation*. Accessed: Feb. 2023. [Online]. Available: https://web3py.readthedocs.io/en/v5/
[108] GitHub. *Repository of the Architecture for the Museum IPR Use Case*. Accessed: May 2023. [Online]. Available: https://github.com/a-moreaux/Lempicka
[109] Ethereum. *Ethereum Virtual Machine (EVM)*. Accessed: Apr. 2023. [Online]. Available: https://ethereum.org/en/developers/docs/evm/
[110] Le Louvre. *Le Louvre Online Tour Portal*. Accessed: Jan. 2023. [Online]. Available: https://www.louvre.fr/en/online-tours
[111] A. Garboan and M. Mitrea, "Live camera recording robust video fingerprinting," *Multimedia Syst.*, vol. 22, pp. 229–243, Mar. 2016, doi: 10.1007/s00530-014-0447-0.
[112] Remix. *Remix Ethereum IDE*. Accessed: Jun. 2023. [Online]. Available: https://remix.ethereum.org/
[113] ISO. *ISO/IEC 23093-1:2022 Information Technology—Internet of Media Things—Part 1: Architecture*. Accessed: Jun. 2023. [Online]. Available: https://www.iso.org/standard/81586.html
[114] ISO. *ISO/IEC 23093-2:2022 Information Technology—Internet of Media Things—Part 2: Discovery and Communication API*. Accessed: Jun. 2023. [Online]. Available: https://www.iso.org/standard/81587.html
[115] ISO. *ISO/IEC 23093-3:2022 Information Technology—Internet of Media Things—Part 3: Media Data Formats and APIs*. Accessed: Jun. 2023. [Online]. Available: https://www.iso.org/standard/81589.html
[116] O. Goldreich and Y. Oren, "Definitions, and properties of zero-knowledge proof systems," *J. Cryptol.*, vol. 7, pp. 1–32, Dec. 1994, doi: 10.1007/BF00195207.
[117] E. Weyl, P. Ohlhaver and V. Buterin, "Decentralized society: Finding Web3's soul," May 2022. [Online]. Available: https://dx.doi.org/10.2139/ssrn.4105763
[118] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, and M. Correia, "Do you need a distributed ledger technology interoperability solution?" *Distrib. Ledger Technol., Res. Pract.*, vol. 2, no. 1, pp. 1–37, Mar. 2023, doi: 10.1145/3564532.
[119] I. A. Qasse, M. A. Talib, and Q. Nasir, "Inter blockchain communication: A survey," in *Proc. ArabWIC 6th Annu. Int. Conf. Res. Track*. New York, NY, USA: Association for Computing Machinery, Mar. 2019, pp. 1–6, doi: 10.1145/3333165.3333167.
[120] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum, Yellow Paper 151.2014, pp. 1–32, 2014.
[121] Ethereum. *Blocks*. Accessed: Jul. 2023. [Online]. Available: https://ethereum.org/en/developers/docs/blocks/
[122] A. O. Bada, A. Damianou, C. M. Angelopoulos, and V. Katos, "Towards a green blockchain: A review of consensus mechanisms and their energy consumption," in *Proc. 17th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, Pafos, Cyprus, Jul. 2021, pp. 503–511, doi: 10.1109/DCOSS52077.2021.00083.
[123] P. K. Sharma, N. Kumar, and J. H. Park, "Blockchain technology toward green IoT: Opportunities and challenges," *IEEE Netw.*, vol. 34, no. 4, pp. 263–269, Jul./Aug. 2020, doi: 10.1109/MNET.001.1900526.



ALEXANDRE C. MOREAUX received the M.S. degree in digital imaging from Telecom SudParis. He is currently pursuing the Ph.D. degree with Institut Polytechnique de Paris. He was a Guest Researcher with the NIST's Network Department. He is also an active contributor to ISO/IEC 21000-23 smart contracts for media and ISO 23093 family—Internet of Media Things.

MIHAI P. MITREA received the Habilitation à Diriger des Recherches (H.D.R.) degree from Pierre and Marie Curie University, Paris, in 2010, and the Ph.D. degree from the Politechnica University of Bucharest, in 2003. He is currently an Associate Professor with Telecom SudParis. His research interests include multimedia content tracking (watermarking, fingerprinting, and blockchain). He is also the Vice-President of the Cap Digital's Technical Commission on Digital Content and serves as an Advisor for the French Delegation at ISO/IEC JTC1 SC29 (a.k.a. MPEG). Inside MPAI standardization organization, he is coordinating neural network watermarking efforts.