Visual content tracking, IPR management, & blockchain : from process abstraction to functional interoperability

Alexandre Moreaux

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAS023

Thèse de doctorat

Visual content tracking, IPR management, & blockchain: from process abstraction to functional interoperability

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Telecom SudParis

École doctorale n°626 de
l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Informatique

Thèse présentée par

Alexandre Moreaux

Composition du jury :

M. Azeddine BEGHDADI Professor, Université Sorbonne Paris Nord	Président du jury
M. Touradj EBRAHIMI Professor, EPFL	Rapporteur
M. Farid MOKRANE Professor, Université Paris 8 Vincennes-Saint-Denis	Rapporteur
Mme. Sabrina CALDWELL Senior Lecturer, Australian National University	Examinatrice
Mme Sara TUCCI-PIERGIOVANNI Head of Laboratory, CEA-LIST	Examinatrice
M. Mihai MITREA Associate Professor, HDR, Télécom SudParis	Directeur de thèse
M. Najah Naffah CEO, Quantum Blockchain Secure	Invité

Everyone, thank you for everything.

Résumé en français

En garantissant un niveau de confiance et d'immutabilité jusqu'alors inégalé, l'émergence synchrone du web3 et des blockchains applicatives a ouvert de nouvelles perspectives dans le domaine du **traçage du contenu visuel** et de la gestion des **droits de propriété intellectuelle** sous-jacents. Cependant, outre les problèmes endémiques à l'aspect décentralisé des **blockchains**, cette association a également mis en lumière de nouveaux défis. La thèse aborde cinq de tels défis à travers le prisme des deux concepts fondamentaux des blockchains applicatives, à savoir les **tokens** et les **Smart Contracts**, et de leur **interopérabilité** (*i.e.*, à leur capacité à partager des données avec d'autres briques applicatives).

Les premiers deux défis relevés concernent les tokens, *i.e.*, la représentation d'actifs sur les blockchains, qui ont fait l'objet de vives critiques de la part des experts et du public en raison de **leurs modes de distribution** et de **leurs lacunes quant aux droits de propriété intellectuelle**. Si une partie de l'aversion envers les tokens résulte d'une incompréhension fondamentale à l'égard de ce qu'ils sont, d'importantes limitantes techniques ont également fait surface au fil des ans. De plus, les tokens ont également été confrontés à des problèmes réglementaires et juridiques qui ont contribué à leur réputation ambiguë.

Les deux défis suivants concernent les Smart Contracts, *i.e.*, les logiciels immuables qui peuvent être déployés sur les blockchains et qui servent notamment aux applications décentralisées (dApps). Au-delà de leur sensibilité aux erreurs humaines, les Smart Contracts sont confrontés à des limitations fondamentales comme **le seuil élevé de connaissances techniques requises** à leur development et **leurs capacités de calcul limitées**. Ces limitations ne sont pas seulement dues à la relative nouveauté du concept mais au fait que la notion de Smart Contract ne vise pas à remplacer celle du logiciel web2 et qu'elles doivent être pensées comme complémentaires.

La thèse répond tout d'abord à ces quatre premiers défis via l'**abstraction de processus** connus afin de concevoir, spécifier, et implémenter des briques méthodologiques répondant à des attentes définies par notre analyse bibliographique du sujet. **Les quatre premières contributions sont :**

- **Un mode de distribution de contenu visuel produit par des objets connectés via un courtier automatique doté de capacités de dépôt fiduciaire basé sur un système de confiance numérique.**
- **Une structure logicielle indépendante des marchés standards mettant les Smart Contracts au niveau conceptuel des tokens afin de garantir l'applications de DPI lors de l'échange de ces derniers.**
- **Un processus permettant la génération systématique et agnostique à l'environnement blockchain de Smart Contracts à partir d'ontologies.**
- **Une méthodologie associant de manière mutuellement bénéfique des éléments web2 et web3 qui permet le calcul d'empreintes numériques (*fingerprints*), dont le coût est normalement prohibitif dans un environnement blockchain.**

Enfin, le dernier défi relevé est celui de **la polyvalence des briques applicatives** que nous aborderons **par l'interopérabilité des quatre premières contributions** dans une architecture permettant la prise en charge de contenu visuel dans l'environnement blockchain, de sa création authentifiée jusqu'à sa distribution tracée et conforme aux DPI. Cette association démontre la capacité de nos briques méthodologiques à être intégrées dans **des workflows complexes**, définis à un niveau **abstrait** tout en répondant à des problèmes tangibles.

Nous concluons cette thèse avec une analyse macroscopique de notre travail, mettant en perspective nos contributions vis-à-vis de du futur des blockchains que nous prévoyons à court et à long terme.

Abstract

Context

Media, and specifically visual content, constitute some of the modern economy's most valuable assets and commodities. From entertainment and social networks to video content generated by meteorological satellites or by cameras for autonomous driving, nearly every applicative vertical benefits from advancements in digital content products and services.

In particular, tracking visual content and applying associated Intellectual Property Rights (IPRs) has been an important part of the technological landscape of the beginning of the 21st century, providing diverse technical and business opportunities. However, these opportunities also present many high-stake challenges from the lack of transparency to online piracy, which have grown tenfold since the advent of the decentralized computing era.

Following their applicative inception in 2015, blockchains have become an essential element of the technological landscape, introducing the concept of web3 (*i.e.*, world-wide decentralized computing) to the public and paving the way for massive investments in distributed computing.

This emergence naturally brought a wide array of natural associations between blockchains and visual content, where the former is expected to provide trust and immutability that is unmatched in the latter conventional solutions, enabling decades of advancements in media technology to leap into the web3 era. Nevertheless, this symbiosis remains theoretical as critical conceptual and technical limitations prevent these associations in practice.

Challenges and limitations

The advent of blockchain applicative ecosystems presented a major shift in web3, bolstering opportunities, challenges, and blockchain skepticism. Critiques of the space began to emerge with the concept of an anonymous, immutable network where complex ideas that are scarcely understood account for or manage a substantial amount of resources. Furthermore, a grey regulatory environment, early abuses, and security breaches contributed to the sentiment of blockchains being unsuitable for licit commercial exploitation. These observations do not even account for technical

shortcomings blockchain infrastructures suffer from[1] (*e.g.,* energy consumption or limited computational capacities).

We address these challenges through the lens of the two fundamental concepts of applicative blockchains, namely **tokens** and **Smart Contracts**, and of **their interoperability** (*i.e.,* as the ability to seamlessly share data). Figure 1 shows the challenges and interoperability limitations we address in this thesis, as detailed hereafter.

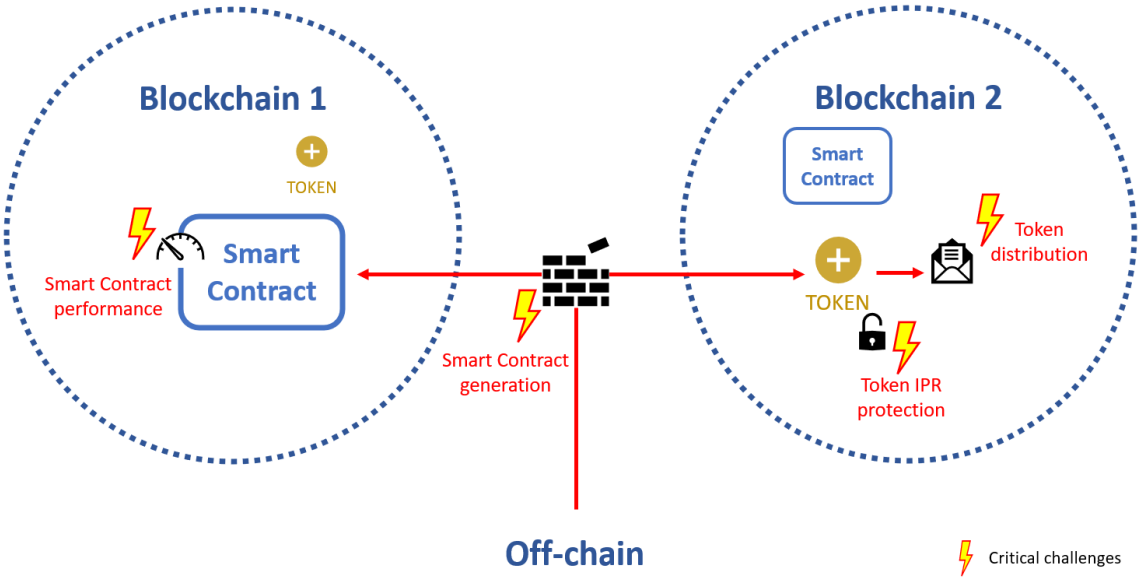

Figure 1: Critical limitations of blockchain applicative technologies and their interoperability. We will focus on four issues, namely: (1) token distribution, (2) the absence of persistent links between tokens and underlying IPR, (3) Smart Contract generation and deployment technical knowledge requirements, and (4) Smart Contract computational performance. Further, we observe that blockchain lack interoperability within a given environment, amongst themselves, and with conventional web2 solutions.

- **Tokens**, *i.e.,* the blockchain representation of assets, have been under heavy fire from experts and the public due to their **centralized distribution** and general **disregard for Intellectual Property Rights (IPR)**. While some issues tokens have faced result from a fundamental misunderstanding of what they are, severe technical limitations in their use and distribution have surfaced over the years. These limitations were also accompanied by regulatory pain points resulting in some of the biggest scandals of the past decade, generating an overall negative perception of web3 technologies.
- **Smart Contracts** are immutable blockchain software that notably power decentralized applications (dApps). Although they have contributed to blockchain scandals through human oversights and errors, their main issues lie in their **technical knowledge requirements** and **limited computation capabilities**. Indeed, the relative novelty of the concept makes these programming languages

[1] Although we introduce the nuances and state-of-the-art of blockchain architecture, this thesis does not address these limitations from a protocol perspective, but rather position itself at the applicative level of blockchains.

less developer-friendly than their web2 counterparts, and users must accept that the Smart Contract paradigm is not a substitute for legacy software.

- **Interoperability** is sometimes identified as the most significant pain point in blockchains. Indeed, only limited data can seamlessly be exchanged between blockchain technological bricks, which limits the **versatility of applicative workflows**. In the thesis, three blockchain interoperability fronts are addressed: interoperability within a given blockchain (*e.g.,* between applications supported on one blockchain *i.e.,* ***intra***-blockchain interoperability), interoperability between blockchains (*e.g.,* the capacity to send information and assets between blockchains seamlessly, *i.e.,* ***inter***-blockchain interoperability), and interoperability between blockchains and web2 software (*e.g.,* in the integration of a high abstraction project, *i.e.,* ***extra***-blockchain interoperability). While each of these can be addressed from regulatory or infrastructure perspectives, we will integrate these notions in forward and backward-compatible applicative solutions answering the previous challenges.

Contributions

This thesis establishes **four methodological frameworks** designed, specified, and implemented to answer requirements stemmed from the gaps identified in the state-of-the-art study. **Two of these methodological frameworks deal with tokens and enable the automatic distribution of assets and the indeterminate, systematic application of IPR (e.g., royalty payments), respectively. The other two primarily deal with Smart Contracts and allow their systematic generation from formalized ontologies and the extension of their computational capacities via a load balancing mechanism, respectively.** We will illustrate these solutions via direct use cases, demonstrating their immediate application.

The **fifth working direction** of the thesis shall combine our newfound methodological frameworks into **higher abstraction architectures that will answer two complex use cases: the tracking of visual content and IPR management in blockchain environments**.

Our solutions will progressively tackle more advanced iterations of these scenarios as they appear throughout this thesis. Further, each of these solutions will integrate at least one of the three interoperability aspects mentioned above. Specifically:

1. **Token distribution**: Our first token-based solution will deal with *intra*-blockchain interoperability by addressing token distribution solutions' lack thereof. It will aim to create a zero-trust systematic distribution means for sensitive Internet of Media Things (IoMT)-produced content via an automatic broker with zero-trust escrow functionalities. We will illustrate this approach through our visual content tracking use case.

2. **Token IPR enforcement**: Our second token-based solution will also deal with *intra*-blockchain interoperability, but from a different perspective. It will focus on the need for enhanced owner control over tokens, especially past first sales. To do so, we designed and developed a zero-trust, rule-enforcing, marketplace-agnostic framework revolving around a paradigm shift putting Smart Contracts at the abstraction level of tokens. We shall illustrate this solution via the IPR management use case.
3. **Smart Contract deployment**: Our first Smart Contract-based solution will tackle *inter-blockchain* interoperability. It will take form in a workflow enabling the systematic, blockchain-agnostic generation of Smart Contracts via formalized ontologies. We directly illustrate this solution via an extension of the IoMT visual content tracking use case. Further, we will explain how we used a similar approach as a part of ISO/IEC JTC1 SC29 WG3 efforts to create an ISO/IEC 21000-23 Smart Contracts for Media standard, where Smart Contracts are generated to fulfill media industry contractual obligations systematically and transparently, thus targeting a further extension of our IPR management use case.
4. **Smart Contract computation capacities**: Our second Smart Contract-based methodology will address *extra*-blockchain interoperability. It will seek to enable a mutually beneficial association of web2 and web3 by allowing usually prohibitively expensive visual fingerprinting to occur in blockchain environments. This approach will enable us to further our visual content tracking use case.
5. **Versatile applicative workflows**: Finally, we will showcase the interoperability of our solutions and integrate all four methodological contributions into an architecture supporting the entire lifecycle of visual assets in blockchains, from their authenticated creation to their tracked and IPR-compliant distribution. We will instantiate this integration for two real-world inspired scenarios: the first will extend our visual content tracking use case in complete IoMT visual data management, and the second will deal with a museum wary of the Intellectual Property (IP) of the visual content they make available online, furthering our IPR use case. Previous notions of both use cases will appear in both these scenarios.

Structure

This thesis is structured into six chapters, which introduce the challenges and context, review fundamental concepts, lay out the current state-of-the-art, present our base methodologies, provide subsequent implementations and their analysis, and conclude this manuscript, respectively.

Chapter 1 contextualizes visual content's opportunities and challenges before putting them in the perspective of the web3 era. These findings will pinpoint the driving forces that power the space and give us a lens to look through for a perspective on the motivations of key actors.

Chapter 2 will begin by presenting the general architecture of blockchains, which the applicative layer relies upon. Specifically, we will review the fundamentals of blockchain infrastructure and transaction structure before delving into modern applicative blockchains. These fundamental notions will ensure that the nuances of our methodology's underlying components are understood before going further.

Chapter 3 will be dedicated to a thorough investigation of the current state-of-the-art of the field beginning with an evaluation of Smart Contracts, token distribution, and token protection. Then, an analysis of current *inter*, *intra*, and *extra*-blockchain interoperability approaches will enable us to identify current gaps in the literature we will address in the rest of this thesis. Finally, we introduce practical concepts from off-chain vertical fields that will be used in conjunction with blockchain in the elaboration of our solutions.

Chapter 4 will bring forth this thesis' methodological contributions. It will begin by formalizing the sets of requirements our work shall follow in light of our state-of-the-art analysis. It will then explain our four base methodologies from a macro perspective. Then, we will provide the synergistic combination of these solutions, highlighting what each can bring to complex high-abstraction solutions.

In Chapter 5, we provide Ethereum and Tezos open-source implementations of the methodologies introduced in Chapter 4 and apply them to relevant use cases, showcasing their contribution to the applicable interoperability front. Then, we will combine all architectural components to offer a complete lifecycle solution for blockchain assets, demonstrating the ability of our technologies to be integrated into larger frameworks. For each of these implementations, we will provide an in-depth analysis of our approach and its performance from a conceptual and practical point of view. We will highlight the most valuable aspects and most significant limiting factors of these solutions, putting them in perspective with the current landscape of web3 visual content and highlighting potential avenues for future work.

Chapter 6 will conclude our work, highlighting our contributions and their place in the state-of-the-art. We shall also use these parting words to gain perspective on the position of blockchains in the landscape of current and future technological advancements.

Summary

The following Table I summarizes the points made in the extended abstract. Issues and challenges are sorted by their relationship with tokens and Smart Contracts. Each will be addressed by an advanced methodology reflecting one of the three aspects of interoperability mentioned above and illustrated via one of two use cases: visual content tracking and IPR management in blockchain environments. These use cases will expand incrementally at each mention to reflect added levels of complexity addressed by our methodologies. A final contribution will see the association of the four previous methodologies to support a combined visual content tracking and IPR management use case.

TABLE I
BLOCKCHAIN APPLICATIVE ISSUES, CHALLENGES, LIMITATIONS, AND THESIS CONTRIBUTIONS

Issues	Challenges	Limitations	Thesis contributions
1. Token distribution	Token exchanges and distribution have become partially centralized around limited use cases and third parties.	Web3 assets have quasi-exclusively been used as representations for digital art, leaving innovations scarce for other applications. Notably, the systematic distribution of protected content in a secure and trusted fashion still requires innovations.	**Methodology:** A zero-trust broker allowing for the distribution of sensitive content. **Use case illustration:** Visual content tracking (IoMT device produced content) **Interoperability front:** *Intra*-blockchain interoperability **Detailed in:** • Chapter 4, Section II.A • Chapter 5, Section II **Disseminated in:** • [ISO23a] [ISO23b]
2. Token IPR enforcement	The link between physical/web2 assets and their blockchain counterparts does not carry IPR properly.	IPR and resulting royalties cannot be enforced systematically, creating an uncertain (at best, hostile at worst) environment for artists and creators where third parties hold the bargaining power due to their reach in the web3 space.	**Methodology:** A flexible framework for the systematic and indeterminate application of rules to tokens. **Use case illustration:** IPR management (systematic and transparent royalty payments) **Interoperability front:** *Intra*-blockchain interoperability **Detailed in:** • Chapter 4, Section II.B • Chapter 5, Section III **Disseminated in:** • [MOR23c]
3. Smart Contract deployment	Smart Contracts have a high knowledge and skill requirement and are at risk of costly errors (in terms of privacy and computation).	The relative difficulty of creating and deploying Smart Contracts makes them a blocking point in use cases requiring highly repeatable workflows with slight variations, which can typically heavily benefit from the reliability and transparency featured by Smart Contracts.	**Methodology:** A workflow that automatically generates and deploys Smart Contracts from specified web2 ontologies. **Use case illustration:** Visual content tracking (IoMT device produced content, Media industry contracts) **Interoperability front:** *Inter*-blockchain interoperability **Detailed in:** • Chapter 4, Section III.A • Chapter 5, Section II **Disseminated in:** • [ALL21a]

TABLE I (CONTINUED)

4. Smart Contract computation capacities	Smart Contracts are seen as the blockchain equivalent of software but cannot power every conventional operation.	Smart Contracts are limited in their computational capacities, available storage, general utility libraries, and quality of life for developers. They were not designed to support complex operations and are still a new field under exploration.	**Methodology:** An architecture enabling the processing of advanced visual feature detection in a blockchain-authenticated fashion. **Use case illustration:** Visual content tracking (near-duplicated content detection) **Interoperability front:** *Extra*-blockchain interoperability **Detailed in:** • Chapter 4, Section III.B • Chapter 5, Section IV **Disseminated in:** • [MOR23a][MOR22]
5. Versatile applicative workflows	Blockchain interoperability limitations restrict the use of blockchain methodologies in broader and higher abstraction architectures and contexts.	Multi-leveled interoperability concerns are one of the most significant barriers to blockchain adoption. Blockchain tools do not systematically integrate well with each other, tools based on different blockchains, and web2 solutions.	**Methodology:** All four previous methodologies are combined into an architecture supporting the complete lifecycle management of blockchain assets created to represent off-chain assets. **Use case illustration:** Visual content tracking + IPR management (IoMT device produced content, Museum visual content protection) **Interoperability front:** *Intra*-blockchain, *inter*-blockchain, and *extra*-blockchain interoperability **Detailed in:** • Chapter 4, Section IV • Chapter 5, Section V **Disseminated in:** • [MOR23b]

Dissemination

Journal papers:

A. Moreaux, M. Mitrea, *Blockchain asset lifecycle management for visual content tracking*, in IEEE Access, vol. 11, pp. 100518-100539, 2023, doi: 10.1109/ACCESS.2023.3311635

A. Moreaux, M. Mitrea, *Visual Content Verification in Blockchain Environments*, in Blockchain and Cryptocurrency, Vol. 1, Issue 1, pp. 44-55, Sep. 2023.

A. Moreaux, M. Mitrea, *Royalty-friendly Digital Asset Exchanges on Blockchains*, in IEEE Access, vol. 11, pp.56235-56247, 2023, doi: 10.1109/ACCESS.2023.3283153

M. Allouche, M. Mitrea, A. Moreaux, S.-K. Kim, *Automatic Smart Contract generation for Internet of Media Things*, ICT Express, Volume 7, Issue 3, 2021, Pages 274-277, ISSN 2405-9595, doi: https://doi.org/10.1016/j.icte.2021.08.009

Conference papers:

A. Moreaux, M. Mitrea, *Blockchain Assisted Near-duplicated Content Detection*, B2C Conference Proceedings, p.98, 2022

Technical inputs for ISO standardization:

A. Moreaux, M. Mitrea

ISO/IEC JTC1 SC29 WG7 m63675 *Modification of data format related to blockchain tokens*, June 2023.

ISO/IEC JTC1 SC29 WG7 m63324 *IoMT NFT*, April 2023.

ISO/IEC JTC1 SC29 WG3 m58473 *MPEG-21 contracts to smart contracts: conversion for the TEZOS blockchain*, December 2021.

ISO/IEC JTC1 SC29 WG7 m58273 *The usage of Tezos DLT solutions for IoMT*, October 2021.

A. Moreaux, M. Allouche, M. Ljubojevic, M. Mitrea

ISO/IEC JTC1 SC29 WG3 m58212 *TEZOS platform for MPEG smart contracts*, October 2021.

A. Moreaux, M. Mitrea, S. Joo

ISO/IEC JTC1 SC29 WG7 m56822 *The usage of DLT solution for vibro haptic devices*, April 2021.

ISO/IEC JTC1 SC29 WG7 m56823 *Advanced usage of DLT solutions for MCameras*, April 2021.

M. Mitrea, A. Moreaux, M. Allouche, S. Joo

ISO/IEC JTC1 SC29 WG7 m57562 *Complex operations computing and blockchain solutions*, July 2021.

P. Kudumakis, M. Ljubojevic, M. Zichichi, M. Allouche, A. Moreaux, M. Mitrea, V. Rodriguez Doncel

ISO/IEC JTC1 SC29 WG3 m56347 *Draft of ISO/IEC 21000-23 DIS SC for Media,* March 2021.

M. Allouche, M. Ljubojevic, M. Zichichi, A. Moreaux, M. Mitrea, V. Rodriguez Doncel, P. Kudumakis

ISO/IEC JTC1 SC29 WG3 m56355 *MPEG-21 SC: Back conversion API* March 2021, March 2021.

M. Allouche, M. Ljubojevic, M. Zichichi, A. Moreaux, M. Mitrea, V. Rodriguez Doncel, P. Kudumakis

ISO/IEC JTC1 SC29 WG3 m56287 *MPEG-21 SC APIs: Back conversion methods*, February 2021.

M. Ljubojevic, M. Zichichi, M. Allouche, A. Moreaux, M. Mitrea, V. Rodriguez Doncel

ISO/IEC JTC1 SC29 WG3 m56157 *MPEG-21 SC APIs: CEL and MCO/MVCO Alignment,* January 2021.

P. Kudumakis, M. Zichichi, V. Rodriguez Doncel, M. Mitrea, M. Allouche, A. Moreaux, M. Ljubojevic

ISO/IEC JTC1 SC29 WG3 m59010 *Text of ISO/IEC FDIS 21000-23 Smart Contracts for Media*, January 2022.

ISO/IEC JTC1 SC29 WG3 m59011 *[13.1] DoC on ISO/IEC DIS 21000-23 Smart Contracts for Media*, January 2022.

P. Kudumakis, M. Ljubojevic, M. Zichichi, M. Allouche, A. Moreaux, M. Mitrea, V. Rodriguez Doncel

ISO/IEC JTC1 SC29 WG3 m56492 *Draft DoC on ISO/IEC 21000-23 CD DC*, April 2021.

P. Kudumakis, M. Ljubojevic, M. Zichichi, M. Allouche, A. Moreaux, M. Mitrea, V. Rodriguez Doncel

ISO/IEC JTC1 SC29 WG3 m56181 *Update to MPEG IPR Smart Contracts WD*, January 2021.

P. Kudumakis, M. Zichichi, V. Rodriguez Doncel, M. Ljubojevic, M. Allouche, A. Moreaux, M. Mitrea

ISO/IEC JTC1 SC29 WG3 m59027 *Draft white paper on 'MPEG Smart Contracts for Media'*, January 2022.

ISO/IEC JTC1 SC29 WG3 m59355 *White paper on MPEG Smart Contracts for Media*, April 2022.

Table of contents

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AVCO	Audio Value Chain Ontology
BaaS	Blockchain-as-a-Service
BFT	Byzantine Fault Tolerance
CBC	Cipher Block Chaining
CEL	Contract Expression Language
CEO	Chief Executive Officer
CNL	Controlled Natural Language
dApp	Decentralized Application
DAG	Directed Acyclic Graphs
DAO	Decentralized Autonomous Organization
DHT	Distributed Hash Table
DLT	Distributed Ledger Technologies
DON	Decentralized Oracle Networks
EEA	Ethereum Enterprise Alliance
EIP	Ethereum Improvement Proposals
E&M	Entertainment and Media
ERC	Ethereum Request for Comments
EVM	Ethereum Virtual Machine
FT	Fungible Token
GDP	Gross Domestic Product
HTLC	Hashed Time-Locks
IBC	Inter Blockchain Communication
ICO	Initial Coin Offering
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IM	Interoperability Mechanism
INATBA	International Association for Trusted Blockchain Applications
IoMT	Internet of Media Things
IoT	Internet of Things
IP	Intellectual Property
IPFS	InterPlanetary File System
IPR	Intellectual Property Rights
ISCC	International Standard Content Code
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
LPoS	Liquid Proof of Stake
MCO	Media Contract Ontology
MPEG	Moving Picture Experts Group
MThing	Media Thing
MVCO	Media Value Chain Ontology
NIST	National Institute for Standards and Technology
NFT	Non-Fungible Token
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
P2P	Peer-to-peer
OECD	Organization for Economic Cooperation and Development
OFAC	Office of Foreign Assets Control
OMI	Open Music Initiative
RM-TLSC	Royalty Management Token Level Smart Contract
R&D	Research and Development
TLSC	Token Level Smart Contract
TPS	Transactions per second
TZIP	Tezos Improvement Proposal
ZKP	Zero-Knowledge Proof

Chapter 1. Introduction

This chapter contextualizes visual content's opportunities and challenges before putting them in the perspective of the web3 era. These findings will pinpoint the driving forces that power the space and give us a lens to look through for a perspective on the motivations of key actors.

1.I. The emergence of visual media

During the last three decades, visual media has become an integral part of the lives of billions of people. Every day, a large portion of humanity reads a news article, shares a picture, watches television, or engages with dozens or even hundreds of visual media instances. Popular knowledge has it that "an image is worth a thousand words," a quote attributed to an advertising executive looking to promote his agency [2]. Although we will not engage in literary analysis or the quantification of advertising impact, it has become a trivial notion that images can convey complex information with uncanny ease. If pressed, we could support this claim by referring to the cultural impact of album and magazine covers, photographs taken during historical events, or logos, to name a few. Figure 2 obliges in a visual panorama of culturally significant images for Western readers born in the 20th century.

Figure 2: Culturally significant 20th century visual media from Western countries.

Further, many industries are now partially or wholly reliant on visual content technologies as part of their processes or as means to communicate internally or with the rest of the world. Every brand, public persona, or identifiable entity now takes great care in fashioning their visual identity. Consequently, visual content has become an omnipresent paradigm for sharing ideas, disseminating information, and shaping opinions. The first visual content revolution occurred during the advent of the user-centric web. This era allowed for the mass distribution and accessibility of visual content, leading to a competitive space producing more impactful media. In this process, many private and

[2] The original advert read "One look is worth a thousand words," and interestingly did not contain any images but 92 words. The saying was rumored to be of oriental origins, which proved to be a marketing strategy. Records seem to show a genesis amongst 1920s Americans [GRA23].

professional assets people owned and manipulated became digital, and visual content became a core of almost every innovative vertical. The current landscape still inherits from this boom over thirty years after the first image was uploaded to the web. The Entertainment and Media (E&M) industry was quantified at almost 2,000 billion USD in 2022 and is anticipated to continue growing [STA23]. However, traditional media is projected to account for a smaller share each year, compensated by the expansion of digital media [STA23], as seen in Figure 3. This figure also shows that the pace of growth has been declining since 2021, a trend that is expected to continue in the foreseeable future. This decrease is explained by [DAV18] not solely by the progressive reduction of traditional media consumption but also by the oversaturation of the market and of users' attention spans, who are now seeking higher-quality content to consume.

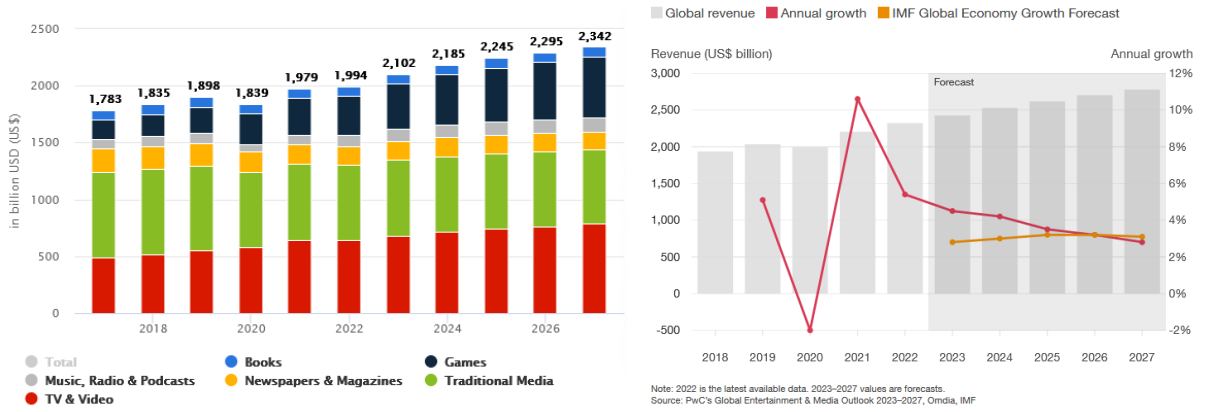

Figure 3: Entertainement & Media global market valuation and projections by Statisa [STA23] (left) and PricewaterhouseCoopers [DAV18] (right).

The surge in media accessibility opened Pandora's box of novel issues content owners and distributors had not imagined and did not have the tools to address. Intellectual Property (IP) and copyright laws were not adequately equipped to deal with this sudden fundamental change in content distribution. Consequently, online piracy skyrocketed to astronomical numbers, which have not dwindled since. The global movie industry loses an estimated 40 to 100 billion USD annually, music pirate sites received 74 billion visits in 2017 alone, and illegal downloading of copyrighted material occupies 24% of the global bandwidth [DAT23]. This traffic's demographic is now spread across generations aged 20 to 50 and geographical zones, with four continents represented in the top 5 offending countries.

The origin of this activity not only stems from the desire for free media but also the lack of convenience and trust in paid options when they exist. A notable disconnect can be observed between regulations and their enforcement, the business workflows and technical tools supporting the latter usually being met with suspicion and reluctance. For instance, digital art still lacks transparency, reliability, and flexibility in enforcing Intellectual Property Rights (IPR). As a result, even the resource-rich music industry suffers from unclaimed royalties that eventually land in a "black box" of money that does not end in the right pockets. The global value of these "black boxes" is estimated between 250 million USD and 5 billion USD [CHR19][RUM17].

Although the media world still suffers from the ripple effects of this first revolution, technological innovations did not wait to provide another significant shift in content distribution. The web3 revolution, carried by blockchains and advocating for decentralization as a core value, rapidly provided new opportunities and amplified the scope of potential issues linked with media.

1.II. Blockchains: a new paradigm

The advent of blockchains marked the shift into the third, decentralized, generation of web paradigms, referred to as web3 [LIU21a]. It follows web2, the social web era where users could actively write information and provide data, which itself followed a more unvarying web1 focused on static content. Web1 only allowed for a limited number of information outlets to be heard. Trust had to be put in said outlets which themselves had to trust network providers. The environment naturally fostered a rustic reputation-based system which lacked a sense of reciprocity [OST03]. The (r)evolution from web1 to web2 was powered by a need and desire of users to be an active part of the overall system, which notably materialized in social media where any user could share multimedia content and communicate with other users around the world.

As such, the Internet became heavily platform driven [COX07]. Although this shift did give some deciding power to users by virtue of making their activity and data commodities, it gave control over the trust dynamics to the service providers running these platforms. Indeed, these centralized dissemination points could control the flow of data to their liking and became *de facto* trusted third parties. Yet, it was not long before that role was challenged. The web2 era gave rise to important questions of censorship and privacy [SHI23] which fueled a discourse of data access models [EYS07]. Notably, it gave cryptography a preponderant position in the flow of data [HAM16a] due to its capacity to put layers of abstraction between data and many its online representations. Yet, all cryptography models rely to some extend on a degree of trust in users or at least in their network identity [BAT13].

This complex, multilayered issue of online trust was addressed from the perspective of a variety of fields including: the social one through the public discourse and debate [AMI19]; the political one in the establishment of a legal framework [WAC19]; the industrial one with the creation of specialized audit groups [SAL19]; or the jurisdictional one in the litigation of policies and post-issue compensation [ROM14]. Importantly, the issue also created a technical niche attempting to create networks that could function without the need for parties to trust each other. This niche eventually gave birth to the decentralized web3. Figure 4 illustrates this simplified evolution of web paradigms.

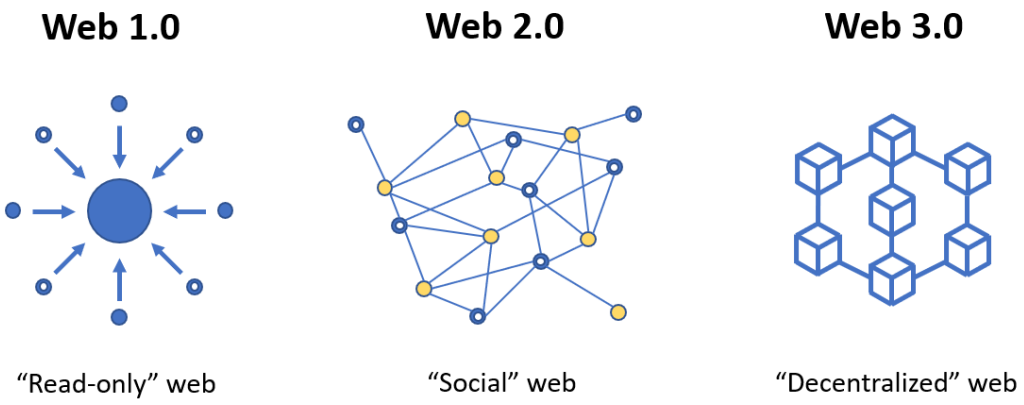

Figure 4: The simplified evolution of web paradigms from the early web1.0 (1991-1999) to the decentralized web3.0 (2014-present).

Web3 aims to incorporate decentralization at the center of web services [MUR23], some examples of which include the metaverse or Decentralized Autonomous Organizations (DAOs). Although the notion, coined by Gavin Wood, is sometimes critiqued as superfluous or a marketing term, its wide use in the industry makes it the *de facto* way to refer to this group of technologies. The first crowning achievement of web3 was the advent of blockchains. Blockchains went from an obscure emerging technology powered by strong ideological stances to a mainstay in the modern industrial panorama in the span of two years. Yet, its repeated appearances in specialized and non-specialized media, scientific and industrial conferences, executive board rooms, advertisements, and even as a day-to-day office conversation topic fail to showcase the depth and nuance of the field.

The notion of *decentralization* has itself shaped into a complex idea full of distinctions. It is defined by Merriam-Webster [MER23] as:

The dispersion or distribution of functions and powers.

Before being a central technological talking point, it was often used in political and sociological contexts (*e.g.,* decentralized decision-making). Put into the blockchain perspective, this definition is often misunderstood and fails to highlight the fundamental notion that centralization or the absence thereof is not a binary feature but a scale, as illustrated in Figure 5. This is particularly important for blockchains that operate on the fundamental postulate that no entity may hold authoritative control over the network. Given that this unspecified entity can be a group as large as one may imagine, practical considerations must limit the scope of this definition to apply it to protocols. Consequently, different blockchains run on distinct protocols with particular levels of centralization, which is a differentiating factor web3 enthusiasts consider before adopting an infrastructure.

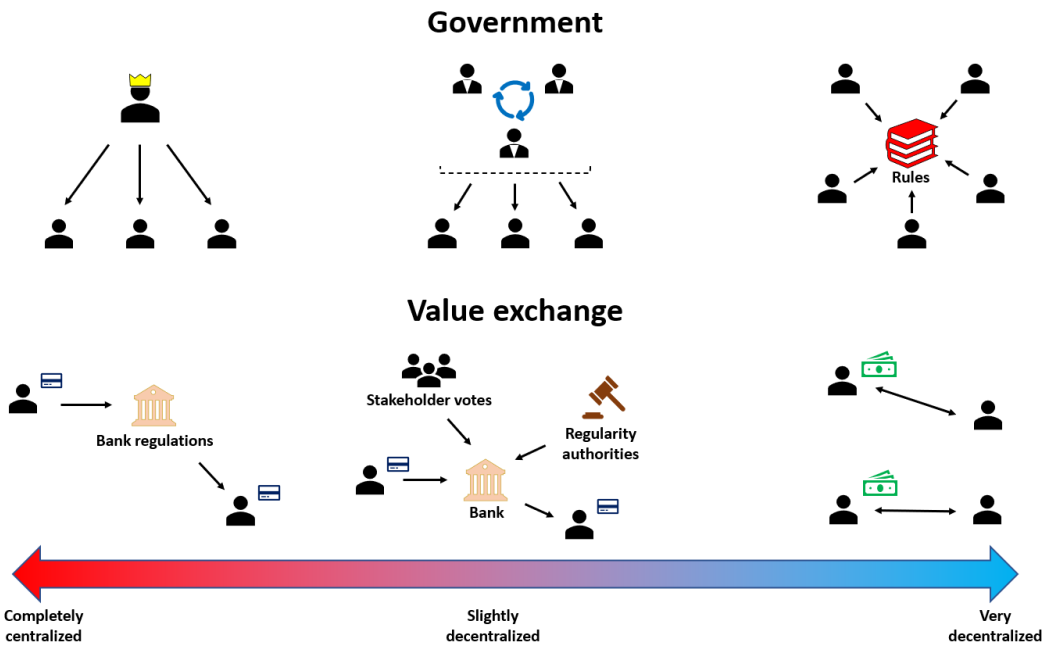

Figure 5: An illustration of decentralization as a scale for government (top) and value exchange systems (bottom). The most centralized schemes revolve around a focused entity (e.g., person, company, institution) while the most decentralized focus on peer-to-peer notions.

In broad terms, blockchains are decentralized, secure, public, permissionless, anonymous networks[3] that enable the exchange of assets and information in a zero-trust fashion. Throughout this thesis, we use the notion of zero trust formalized in [NST20], *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" [BUC21]. The first blockchain, Bitcoin, was designed on cryptographic foundations and brought forth in a 2008 white paper under the pseudonym Satoshi Nakamoto to exchange cash without a central authority's intervention (*e.g.,* a bank, government, *etc*.). The concept was further equipped with applicative capabilities formalized in a white paper marking the start of the Ethereum blockchain [BUT14]. The field is technically and conceptually complex and requires heavy base knowledge before discussing advanced aspects. As such, we dedicated Chapter 2 to explaining the technical fundamentals, applicative pillars, and the historical faults and limitations of the field. The chapter is designed to catch up unfamiliar readers and deepen all reader's knowledge of the infrastructure as the rest of this thesis will make heavy use of the presented notions. It will also enable us to lay a technological portrait of the space at the time of writing, highlighting the events that led up to the modern landscape.

Regardless of their technical specificities, blockchains innovated a wide range of conventional businesses, including finance [CHE20], auditing [LOM22], and IPR management [RAM21], to mention but a few. Their market penetration accelerated around 2016, when the technology reached early adoption in financial services with an adoption rate of 13.5% [KA21a] [WOO17]. 2016 also saw the creation of the Global Blockchain Forum by the Chamber of Digital Commerce to shape policies and establish best practices to streamline jurisdictional friction and facilitate the integration of blockchains in industry [CHD16]. Since then, blockchains have exploded in popularity, and use cases carried by prominent private players now include banking, telecommunications, healthcare, energy, retail, and manufacturing. The leading blockchain activity remains peer-to-peer financial services, grouped under decentralized finance (DeFi). The global blockchain market surpassed 10 billion USD in 2022 [GVR23a] and is projected to reach over 450 billion USD by 2030, with an estimated compound and growth rate of 59.9% from 2023 to 2030 [FBI23], reaching as high as 86.2% for the US market [GVR23a], as shown in Figure 6. Collaborations between various activity sectors and between private and public actors hint that new opportunities are still being pursued actively.

[3] Each of these characteristics is not universal across blockchains, which are not even networks *per se*. We explain all the nuances of blockchain infrastructure in Chapter 2.

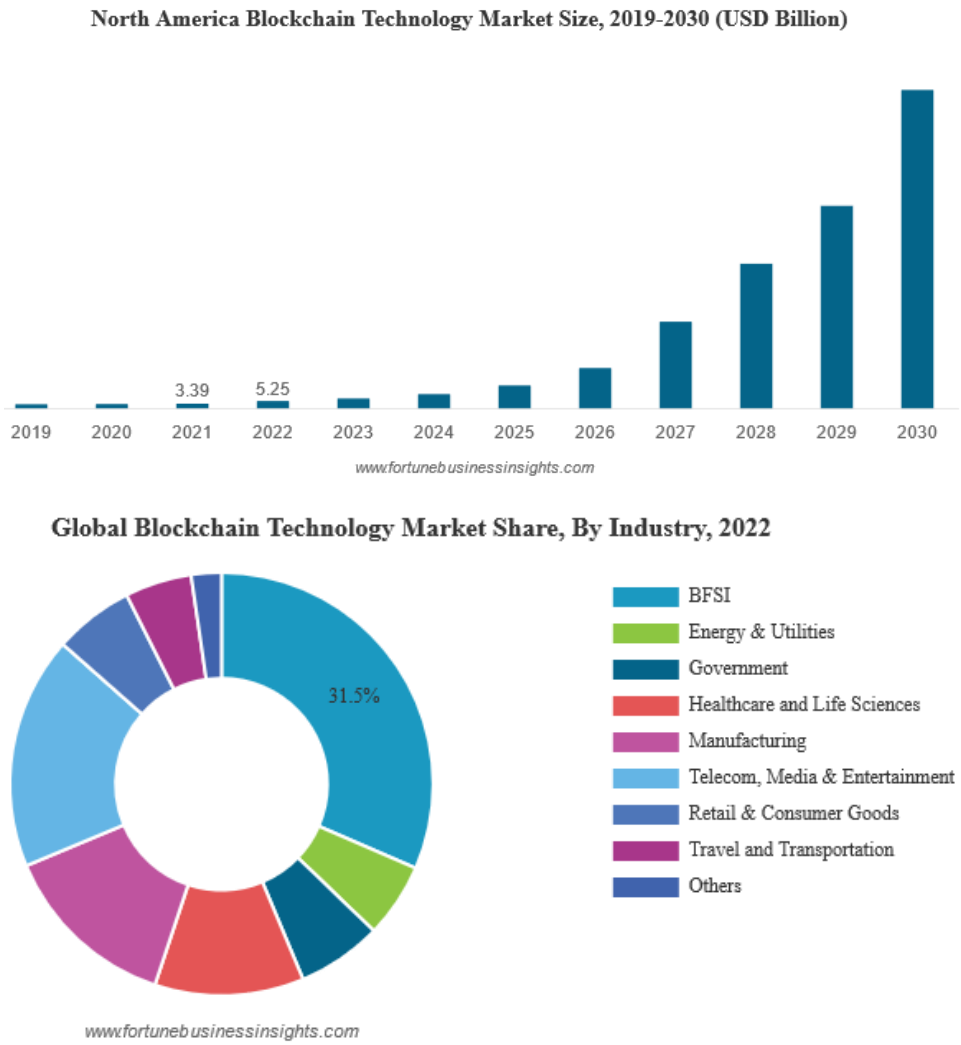

Figure 6: North America blockchain market values and projections (2019-2030, top) and global blockchain market share by application in 2022 (bottom) as per Fortune Business Insights [FBI23].

Despite their successes and perspectives, blockchains must face many challenges, two of which we find most significant. The first, which we will not deal with in this thesis, lies in the hardships of regulating blockchains. Blockchains are designed as cross-border anonymous networks, making regulatory demarcations (*e.g.,* taxation) ambiguous. Collaborations of blockchain entities, governmental agencies, industries, and specialized consortiums are actively tackling these issues. The second, which shall be a cornerstone of our approach, is interoperability. With the variety of blockchain infrastructures, protocols, and applicative prospects, using various blockchain solutions simultaneously or within complex workflows is challenging. This could also be observed with the simple transfer of assets blockchains were designed to provide. Specifically, within the context of computer systems, *interoperability* is defined by the Oxford dictionary [OXF23] as

The ability of systems or software to exchange and make use of information.

This definition was initially built around information technology, but it can apply to blockchains in three lights:

- The capacity for blockchain software (as in, multiple pieces of software living on the same blockchain) to exchange and use information.
- The capacity for blockchains (as in networks/protocols) to exchange and use information.
- The capacity for blockchains to exchange information with non-blockchain systems or software.

Although one might understand that blockchains built for different purposes do not necessarily interoperate, the other points might be startling. Why would blockchain applications built towards the same goal not interoperate? In a word, competition. Blockchain traffic is a valuable resource that web3 business-to-customer actors would often rather not share, *e.g.,* heavily (*de facto*) standardized blockchain assets created on some platforms are sometimes deliberately not compatible with others (cf. Chapter 3, Section 4), as a client lock-in business strategy.

The reasons for limitations in blockchain connections with web2 software are methodological rather than economic. These connections are particularly relevant in blockchain applications, which can get and send back information to legacy systems within certain constraints. Yet, the relative novelty of the programming languages supporting these applications limits their capabilities and quality-of-life features. Consequently, developers have not uniformly adopted these languages and methodological bridges, which have remained centered around specific necessities. Furthermore, the intrinsic limitations of web3 software which runs on decentralized networks, make standard software integration an often-inconceivable approach.

1.III. Linking media and blockchain

The assimilation of media concepts in the web3 space was one of the highlights of the applicative blockchain evolution. Immediately, the potential for decentralized media assets to be distributed on a large scale crystalized in the emergence of a custom digital asset (Non-Fungible Token, cf. Chapter 2) market reaching 44 billion USD in 2021 [CHAIN1] [CHA22b] on the Ethereum blockchain, dwarfing the cryptocurrency market size (2.3 billion USD in 2022 [EMR23] [BRC23]). This success prompted heavy investments in high-abstraction media verticals for blockchains. These investments have now painted the idea of a future of user-interfaced decentralized software, the first instances of which already exist.

In particular, social media and video games are high-perspective fields for decentralized computing which have seen their first implementations. [ALC23a] provides a list of web3 social media platforms, and [ALC23b] lists web3 video games. Although these instances do not come close to competing with the leaders of their industries, their variety and the interest of industry giants show the field's activity and the actors' willingness to allocate resources to the field. This interest materializes in R&D investments, *e.g.,* blockchain games attracted 739 million USD in investments in the first quarter of 2023 [DAPP23].

Yet, these impressive figures are tainted with prospective stolen content, money laundering, outright scams, and wash trading aimed at artificially increasing the value of assets. Briefly:

- The monetization of content by non-IP holders is all too common and facilitated by the low level of risk associated with the activity. It is difficult to pursue any legal action if a fraudster is even identified, which pairs with the general misunderstanding of blockchain assets and their associated rights (cf. Chapter 2). A famous case involved a piece of art by Banksy being sold by another person as a blockchain asset (as reported by media outlets [TID21] [BAK21]). The most popular blockchain platforms have created moderation teams tasked with taking down plagiarized content, but their effect has been limited by the sheer quantity of volume to be inspected.
- The ease of trading and subjective pricing of assets makes them the perfect vessel for money laundering funds originating from fraud, hacks, *etc*. Further, dedicated services called crypto mixers or tumblers exist to obscure the provenance of funds by grouping and redistributing funds from diverse sources for a fee. According to [CHA22c], over 26 billion USD in cryptocurrency were laundered as such between 2019 and 2021, which does not include fiat currencies (*i.e.,* currencies not backed by commodities, including but not restricted to legal tenders *e.g.,* euros or USD), or physical commodity earnings (*e.g.,* gold).
- Many web3 initiatives have proven to be scams targeting investor money. A base of building expectations via celebrity endorsements and large-scale marketing campaigns, a healthy dose of insider trading (usually via pump and dump strategies), a pinch of *greater fool theory* [VLE17], and the sudden disappearance of a company having generated fortunes in investments has

been a popular recipe in blockchains with examples too numerous to all mention here. Blockchain-based Ponzi schemes have also repeatedly made mainstream news boasting stratospheric figures (*e.g.,* Bitconnect [DOJ22a], OneCoin [DOJ23], or PlusToken [COI20a]), with the most famous leading to the arrest and federal indictment of Sam Bankman-Fried, the ex-CEO of the FTX cryptocurrency exchange, on securities and wire fraud charges [DOJ22b].

- [CHA22c] analyzed self-financed wallets making asset purchases on the Ethereum blockchain and found a group of 110 traders who made almost 9 million USD in profits, pointing out that their calculations were undoubtedly underestimations due to the scope of their study. This last point is not only a concern at the level of individuals attempting to misrepresent the demand for their assets but at the level of platforms with incentives to boast impressive trading volumes.

As with many things in blockchain at the time of writing, these activities fall in unregulated gray zones which are yet to be engaged with by web3 communities and regulatory agencies. It is also important to note that blockchain did not create large-scale fraudulent activity as much as redirected part of fraudulent traffic and that the figures, although impressive, pale compared to similar issues outside blockchains. For instance, the UN Office of Drugs and Crime estimates that between 800 billion USD and 2 trillion USD are laundered annually [UND23]. These values correspond to 2 and 5% of the global GDP, while blockchain money laundering figures are closer to 0.05% [CHA22c]. The famous saying “When there’s a will, there’s a way” applies particularly well to money laundering, which is a part of all asset exchanges.

Paradoxically, many of these issues are easy to spot due to the transparency of transactions, which unfortunately does not help further investigation. Additionally, blockchain investigation remains a tedious activity bolstered by the ease for fraudsters to create multiple accounts, not to mention that these accounts are not tied to real-world identities, giving a definite edge to malicious actors. Private and public entities have nonetheless taken countermeasures via the emergence of blockchain detective companies and agencies and the inclusion of blockchain interest from government regulators. For instance, the Office of Foreign Assets Controls (OFAC), a part of the US Treasury, has imposed Specially Designated Nationals (SDN) blocks on several blockchain services [TRE21] (*e.g.,* Chatex , a P2P exchange known for accepting fraudulent funds).

1.IV. Summary

In short, media innovations of the previous decades do not present inherent methodological contradictions with web3 concepts. Additionally, web3 paradigms bring forth data robustness and distribution levels which are novel to media technologies. Yet, such absence of theoretical blocking points does not automatically imply immediate symbiosis. The web3 paradigm has exacerbated issues media-related industries have been fighting for decades and created new ones. These issues include but are not limited to, stolen content, royalty hijacking, IPR infringements, *etc*. The emergence of joint solutions is limited by:

- The difficulty of conceptualizing the joint use of web2 and web3 paradigms, *i.e.*, **process abstraction**.
- The lack of methodological bridges enabling implementation, *i.e.,* **functional interoperability**.

Although media-related technologies are vast, overlaps with blockchains are concentrated around specific use cases. We shall attempt to provide the entire life cycle of media assets on blockchains by examining (1) the use of advanced media detection in blockchain environments, (2) the blockchain representation of media assets and their relationship with underlying physical or software assets, and (3) the distribution of blockchain media representing assets. This process is illustrated in Figure 7. Although not exhaustive, these steps encapsulate the fundamental uses of blockchain media assets, from their creation to their distribution, while also considering their link with outside paradigms.

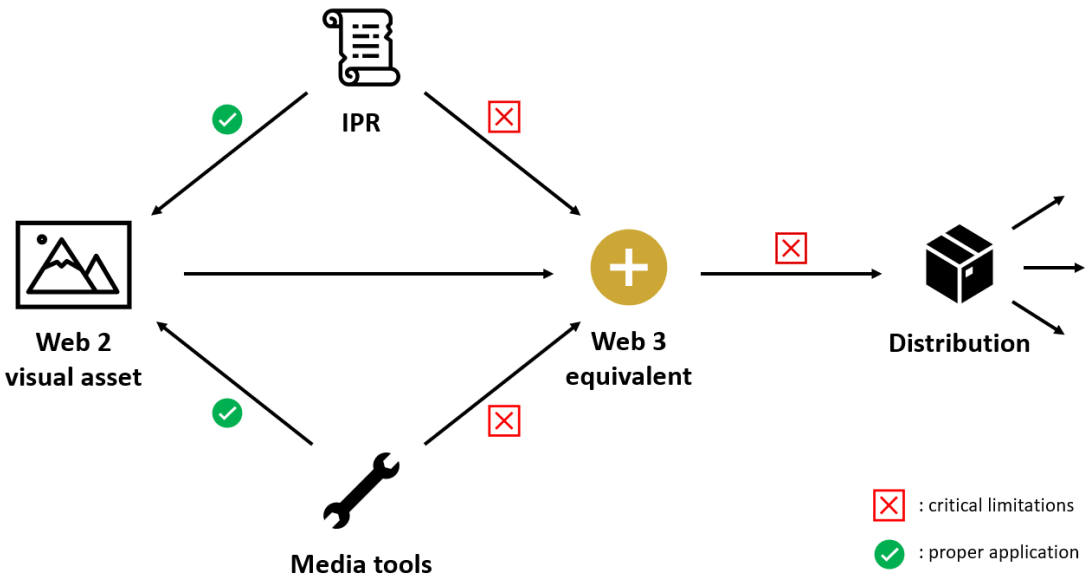

Figure 7: Simplified overview of the lifecycle on web2 visual assets in web3, showing critical limitations.

These steps will fundamentally rely on two technologies: **visual content tracking**, which enables us to maintain a link between a blockchain asset and its physical or web2

counterpart, and **IPR management**, which ensures the good standing of the asset and formalizes underlying rights, notably in its distribution. These two questions will consequently be at the center of the practical use cases we shall tackle. Specifically, we will focus on providing methodologies, governance mechanisms, and software targeting the joint use and the mutual benefit of web2 media and web3 **blockchain** technologies through the lens of interoperability concerns. Security mechanisms shall not be part of our research scope. Instead, we inherit from the native features of blockchain infrastructures and the current state-of-the-art of web3 security and data robustness (cf. Chapter 2 and Chapter 3).

After establishing a solid foundation on blockchain fundamentals and the field's state-of-the-art in Chapters 2 and 3, we introduce four solutions targeted at gaps in the state-of-the-art in Chapter 4, which we design under the constraint of requirements identified in our analysis of the space. We then apply these methodologies to direct use cases before combining them to address more complex issues, thus demonstrating their interoperability and complementarity. Chapter 5 will provide the technical implementation of these solutions and their combination, but also analyze their advantages, limitations, and avenues for future work. We close this manuscript with a retrospective view of our work and perspective on the subject in Chapter 6.

Throughout Chapters 2, 3, and 4, we will emphasize important points using Blockchain Basic, Key Issue, Critical Limitation, Design Requirement, and Methodological Solution frames thusly:

Blockchain Basic: These frames be used in Chapter 2 and Chapter 3 and will highlight an important piece of information related to blockchains and ensure common definitions.

Key Issue: These frames will be used in Chapter 3 and will highlight key issues we address throughout this thesis (as summarized in the *Issues* and *Challenges* columns of Table 1).

Critical Limitation: These frames will be used in Chapter 3 and will highlight the identification of a critical issue we will address in our methodologies (as summarized in the *Limitations* column of Table 1).

Design Requirements: These frames be used in Chapter 4 and will list the requirements we impose based on our state-of-the-art findings.

Methodological Solution: These frames be used in Chapter 4 and will provide a summary of one of our solutions (as shown in the *Thesis Contributions* column of Table 1).

Chapter 2. Blockchain in a nutshell

This chapter will begin by presenting the general architecture of blockchains, which the applicative layer relies upon. Specifically, we will review the fundamentals of blockchain infrastructure and transaction structure before delving into modern applicative blockchains. These fundamental notions will ensure that the nuances of our methodology's underlying components are understood before going further.

2.I. Blockchain architecture

2.I.A. Fundamentals

A blockchain is a distributed ledger supported by a peer-to-peer network[4] of nodes[5]. It belongs to the aptly named family of Distributed Ledger Technologies (DLTs). Not only does the notion of a shared trusted ledger date back millennia but so does the idea of distributing said data record across multiple locations or "nodes", which has re-occurred throughout time and cultures. Historical examples include the Roman Empire's banking system [TEM04], Qing Piaohao draft banks [WIL16], or Yap Island Fei stones [FIT03] as early as 500 BC, each bearing their own specific tangible records, methods of distribution, and anti-tampering systems. The beginning of the modern era in DLTs began in 1991 with [HAB91] putting forth a document timestamping method which eventually led to Bit Gold [SZA05], a digital currency not reliant on a trusted third party. File sharing systems relying on distributed networks were then developed and paved the way towards blockchains [NAK08].

It is important to note that although the terms "blockchains" and "DLTs" are used interchangeably in many forms of media and the exact semantics are filled with grey zones still being debated, blockchains are a subset of DLTs. Other examples of DLTs may use different data structures, cryptographic means, and integrity mechanisms although they have the same aim of providing secure, decentralized communication. Some examples include IOTA [IOT22], an Internet of Things specialized DLT based on Directed Acyclic Graphs (DAG) [HEL21]; Hashgraph [HED18], a scalability focused DLT running a "gossip protocol" consensus mechanism; or Holochain [HOL17] where nodes process their own ledgers using a Distributed Hash Table (DHT) [HOL23]. Our focus on blockchains is led not only by the resource availability and initiative frequency on blockchains but also by the use cases we tackle hereafter (*c.f.* Chapter 5).

The first blockchain was brought forth in a 2008 white paper by Satoshi Nakamoto [NAK08] (a pseudonym yet to be tied to a person or group), based on zero-trust notion implementations and immutable timestamp chains [NAR16] [HAB91] [BAY93]. A blockchain periodically generates a block of data appended to the previous via cryptographic hashes, hence creating a chain of blocks iteratively linking every block up to the first or genesis block. This cryptographic link ensures no data block can be tampered with without affecting all previous blocks, a feature provided by the Merkle (or Binary hash) tree structure and ensuring high Byzantine fault tolerance (BFT, [DRI03]). In theory, blockchains can sustain 50% BFT in theory, but social coordination attacks make it 33% in practice [BUC16]. This link is ensured by Cipher Block Chaining (CBC)[BEL94] and a reoccurring Merkle root [CAS21] between block header, and is illustrated in Figure *8*. Given that the information contained in the blocks is updated on every node, not only does a node going down not affect the history of recorded data, but a single node remaining functional will ensure the perennity of the data. This feature makes most

[4] The semantic shortcut referring to "blockchain networks" is common, and we shall oblige with it for conciseness' sake.
[5] This definition is directly inherited by [NAK08], which introduced the concept to the world.

blockchains "eventually consistent" rather than consistent (*i.e.,* at any given time, all nodes in the network have the same data) by Brewer's CAP theorem [BRE01], limiting data storage features to two of three properties amongst Consistency, Availability, and Partition tolerance [BRE12]. The tradeoff is the necessity to process the data on every node. Moreover, the permissionlessness and anonymity of the network make it a zero-trust one, upholding the "never trust, always verify" policy. Indeed, no intrinsic trust between actors is required in the blockchain framework, which is powered by collective self-interests.

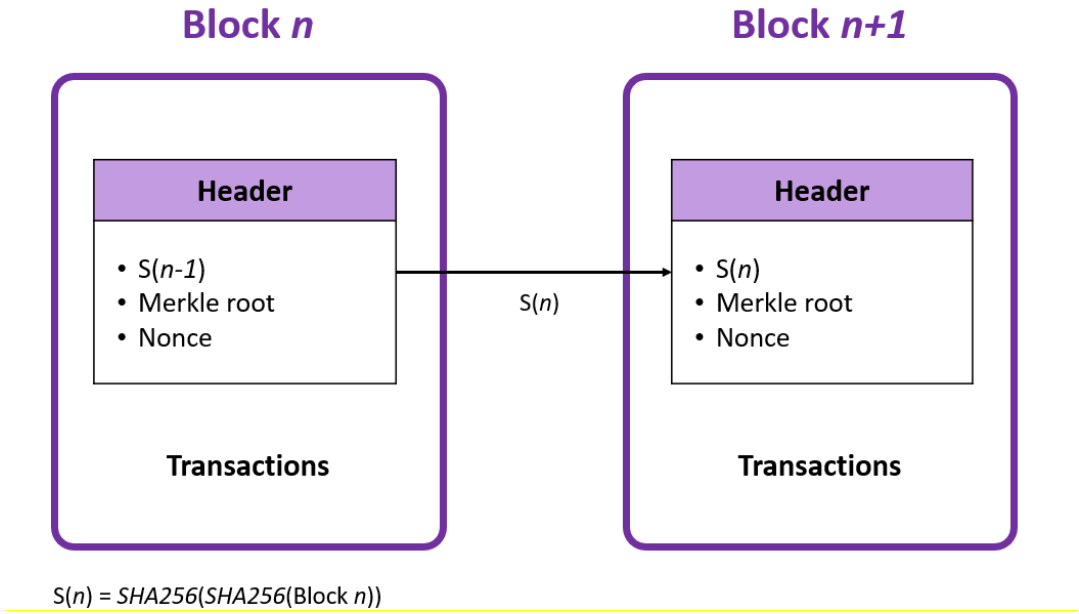

Figure 8: Blockchain block construction, with a focus on block headers and the cyptographic link they form.

Blockchains were conceptualized as public and permissionless, allowing anyone to open a node or to read and write on it. High energy consumption and complete openness prompted alternative philosophies to emerge, which birthed private and permissioned blockchains. Private blockchains verify users' identities before inviting them into the network, and centralized parties can decide to block or modify any data. Many blockchain enthusiasts have criticized private blockchains, sometimes questioning their etymology as they are entirely centralized. Yet, the model has seen adoption as a secured, shared database for organizations and some of the most widely *test chains*, where developers can experiment without fear of losing their assets. This is supported by the fact that these blockchains are much faster and easier to run. Their widespread use as such makes them, *de facto*, blockchains. Permissioned blockchains provide a middle ground, where users can join the network after an identity verification process and are consequently attributed a role (*i.e.,* a set of permissions). This model is particularly well suited to Blockchain-as-a-Service (BaaS), which allows businesses to use blockchains for targeted needs without heavy infrastructure or resource investments. This comes at the expense of requiring web2 Internet connections vulnerable to hacking. For completeness' sake, the sidechain is another commonly found type of blockchain. Sidechains are independent chains with a protocol, consensus, *etc*., linked to a main blockchain (mainchain) using a two-way bridge. We shall discuss them in more detail in Chapter 3.

Blockchain protocols are sometimes updated, the logistics of which largely depend on the protocol in question. Most require forks, which provide minor updates in a backward-compatible fashion, or sometimes hard forks, which are new, backward-incompatible changes that require users to upgrade their software. The former enables patching up newly found issues or improving quality of life, while the latter is used when a significant philosophical debate cannot be settled, leading to multiple versions satisfying all parties (cf. Section III).

Blockchain Basic: Blockchains are protocols supported by P2P networks. They provide a resilient, anonymous, decentralized, permissionless, zero trust, and immutable ledger of transaction data.

In addition to hashes, blocks contain transaction data and timestamps. These transactions are signed data packages sent between users through anonymous accounts referenced by addresses supported by public key cryptography and allow the exchange of the native cryptocurrency supported by the blockchain, which is referenced via three-letter symbols (*e.g.,* BTC, ETH). These accounts are not tied to any identity and effectively belong to whoever possesses their private key (a famous blockchain adage goes, "Not your key, not your crypto."). Many third parties offer wallets that can manage multiple accounts. Hot wallets can connect to the Internet, while cold wallets are hardware devices that remain offline.

Blockchain Basic: Parties are identified through accounts found via addresses and belong to their private key holder.

2.I.B. Transactions and consensus

To run transactions, the user's requests join a transaction pool where the transaction is to be validated before being included in a block. Validators[6] are decided upon using a *consensus algorithm*. They are responsible for ensuring transactions are not fraudulent, for which they are paid *gas* and fees. Gas represents the value of computational power in blockchain and takes the form of a fee paid out to execute transactions on the network. When requesting a transaction, a user will decide the gas limit they are willing to spend on the transaction and the price they are willing to pay for said gas. Higher gas prices will incentivize validators to treat a transaction in a priority. These values are now estimated automatically in most cases, as they depend on the congestion of the network.

A fundamental aspect of blockchains is that reading information must be fast. As such, whether an operation writes on the blockchain constitutes a critical distinction: a transaction writes data on the blockchain, must be validated, and costs gas; an operation such as checking a balance does not constitute a transaction and is consequently fast and does not cost gas.

[6] The term *validator* comes from Ethereum but is commonly used to refer to similar roles in different blockchains ,*e.g.,* Tezos *bakers* [TEZ23b].

Blockchain Basic: Read-only operations are fast and free, while transactions are slow and cost gas.

The validators that ensure transactions are not fraudulent are elected through consensus algorithms, the most widespread being:

Proof of work (PoW): the historic blockchain consensus mechanism based on Hal Finney [NAK04] that elects validators by making them spend computational power to solve for hashes (*i.e.,* find a SHA-256 nonce [CRY22]) in a *brute force* fashion. The first *miner* to find the key to this cryptographic problem bears the role of validator for the block. In case of simultaneous findings, the longest chain principle (equivalent to the [LEA19]) applies. This scheme incentivizes more miners to attempt to solve the problem, making the problem harder to solve and hence the network more secure. Indeed, the computational power required to affect the network becomes impractical due to the cumulative sunk cost that an attack would require. A 2021 study [LEA19] put the conservative cost of a 51% attack[7] on the then Bitcoin network at USD5.5 billion. It would be more profitable to participate in the protocol. This strategy also naturally regulates the creation of cryptocurrency by nature of the difficulty adjustment of the cryptographic problem implemented by PoW blockchains. The flip side of this feature is that PoW networks are extremely power-hungry (notably because of the duplicative process), which makes their scalability limited and their environmental impact significant. This is one of the key factors prompting the shift towards Proof of Stake. Additionally, this led to users with limited power joining mining pools which would distribute the reward if the pool were to validate, somewhat centralizing the process.

Proof of Stake (PoS): in this system, validators stake a portion of their assets in exchange for a chance to be elected to validate transactions. Their validation is subsequently attested by other validators, after which the network can update the blockchain and reward participating validators with a fraction of the fees, divided proportionally to the amount staked and the length of time it has been staked for. Errors in validation or node downtime result in the *slashing* of the stake. Similarly to PoW, the system is resistant to 51% attacks by design, as the loss of the required stake would outweigh potential rewards. This validation method does require technical knowledge and a significant stake. Consequently, staking pools run by trusted validators have emerged, enabling users with limited technical knowledge to earn fractions of fees by contributing to the stake of a pool. Some of the main drawbacks of PoS include the fact that wealth is directed towards those who can stake the most assets, *i.e.,* "the rich get richer"; the hedging of bets from potential validators, which is hard to penalize fairly ("nothing at stake"); or even the hardships of a reliable pseudo-random generator. These issues have spawned variations on the PoS idea, notably Delegated Proof of Stake (DPoS), which

[7] A 51% attack refers to the idea of gaining control of the network by possessing a non-over rulable portion of the available resources. Details on the blockchain iteration of the notion are given in upcoming subsection.

addressed the "rich get richer" issue by giving users a vote towards validators who split their earnings [LAR14] (with many variations [YAN19][HU21]) or Liquid Proof of Stake (LPoS)[TEZ23d], which lets users loan their validation rights while retaining ownership of assets linked to said right [ALL19].

Proof of Authority (PoA): a less traditional consensus algorithm that designates specific machines beforehand and allows them to validate transactions. Validators are clearly identified, and their reputation is tied to their identity [MAN22a]. Specifics vary in the unanimity, majority, or single validator that needs to approve transactions. This scheme enables validators to prevent certain transactions without consequences, which could bring an element of censorship to the network. To prevent this, validators with conflicts of interest and no mutual trust are often instated, which leads to each validator closely monitoring the actions of others and acting in the best interest of the stability of the network. Nonetheless, this structure undoubtedly undermines decentralization in the compromise to become faster and more energy efficient (as there is no direct competition between validators). This enables such blockchains to bear faster *block rates* (block generation rates) and sometimes do not even support a native cryptocurrency. PoA blockchains are typically private blockchains, although public PoA blockchains do exist.

These mechanisms have their own variations (*e.g.*, "Liquid" or "Delegated" PoS which we will discuss further) and are far from the only consensus methods in the field. We can also mention Proof of Burn, Proof of Activity, Proof of Importance, Proof of Activity, a variety of practical applications of Byzantine fault-tolerance algorithms, or any other of the dozens of methods aiming at establishing trust amongst *a priori* untrustworthy actors. A survey of methods and a comparison of their respective performances can be found at [XIO22]. An analysis of these consensus algorithms' history and future can be found in [BAM20], [AZB21], and [BAC18]. Importantly, this thesis' contributions are agnostic with respect to the specific validation process of the blockchain in context.

Once the consensus mechanism has elected a validator, a digital signature of the block upholds the integrity of the subsequent validation. Block finality (the certainty that a block will not be modified or excluded from the chain) is rather probabilistic than binary, as the more blocks are generated after a given block, the more blocks a potential change must affect. This links back to the "eventual consistency" imposed by the CAP theorem.

Blockchain Basic: Blockchain operations are *validated* by consensus algorithms before they are appended to a block.

2.I.C. Examples

The historic blockchain, Bitcoin, is public and functions on PoW and was created solely to support its cryptocurrency (BTC). It supports block sizes up to 1MB (corresponding to approximately 2000 transactions) and targets to generate blocks every 10min. This size in MB is a thing of the past, as a protocol called Segregated Witness (SegWit, [SIN20]) replaces the concept of block size with block weight. This makes Bitcoin's effective block size limit closer to 4MB. A fork of Bitcoin called Bitcoin Cash elected to go its way to support more transactions per minute, raising their block size limit to 8MB (32MB using SegWit). BTC is also limited in supply, as only 21M BTC will ever be mined (actually, 20,999,999,9769 due to how numbers are handled). This tends to increase the demand and price, coupled with the fact that the remaining BTC will be harder to mine (acquire, in this context). The last BTC is estimated to be generated in 2140 due to Bitcoin's protocol halving miner rewards every 21,000 blocks, which renders the cryptocurrency scarcer with time. For instance, the first Bitcoin blocks (2008) rewarded miners with 50BTC, while blocks in the latest halving phase (which will last until April 2024) are rewarded 6.25BTC for mining a block. This will not kill the blockchain, as Bitcoin's main appeal is to store value, and a decrease in miners will increase the price of operations, as per the self-adjusting design of the chain dictates.

Since the advent of Bitcoin, many new blockchains have emerged. The most successful have often relied on massive Initial Coin Offerings (ICO), where future cryptocurrency is promised to raise capital. These ICOs accounted for billions of dollars of investments toward blockchains from actors attempting to speculate on the future value of assets they were supposedly buying at a discount. Two blockchains with the largest ICOs for their time, Ethereum and Tezos, will constitute the infrastructure supporting our work in the field:

- The now largest blockchain, Ethereum, not only supports a cryptocurrency of its own (ETH) but also jumpstarted the application-enabled era of blockchains (cf. Section II). As such, its block rate is much higher at 1 per 15 seconds and replaced its block size limit with a gas limit of 30 million gas. This approach ties into the self-regulating aspect public blockchains target. Ethereum began as a PoW blockchain (Ethereum 1.0), but limitations mentioned in the previous section prompted a switch towards Ethereum 2.0 beginning in December 2020, culminating in a merge in September 2022, which runs on PoS (with a minimum stake of 32ETH). This moniker encompasses several improvements (notably related to network congestion and scalability, which sometimes caused very high gas prices) but did not rename the blockchain, still referred to as Ethereum. After this change, energy consumption was lowered by a drastic 99.95% [ETH23a], and the block rate is now closer to 1 per 12 seconds (supporting approximately 29 transactions per second (tps)[8]). Figure 9 compares Ethereum's former and new energy consumption with Bitcoin and other widespread industries. It shows that Ethereum 2.0 now uses eight times less annual energy than AirBnB, and 50,000 times less energy than

[8] Ethereum claims this system will enable to support 100,000 tps in the future [LED22].

Bitcoin which remains on par with the gold mining industry [ETH23a]. These major changes are decided off-chain and are discussed and voted on by a panel of core members (cf. Section II.B). Ethereum's ETH supply is unlimited, which will very likely cause the financial value of the cryptocurrency to decrease as more are issued.

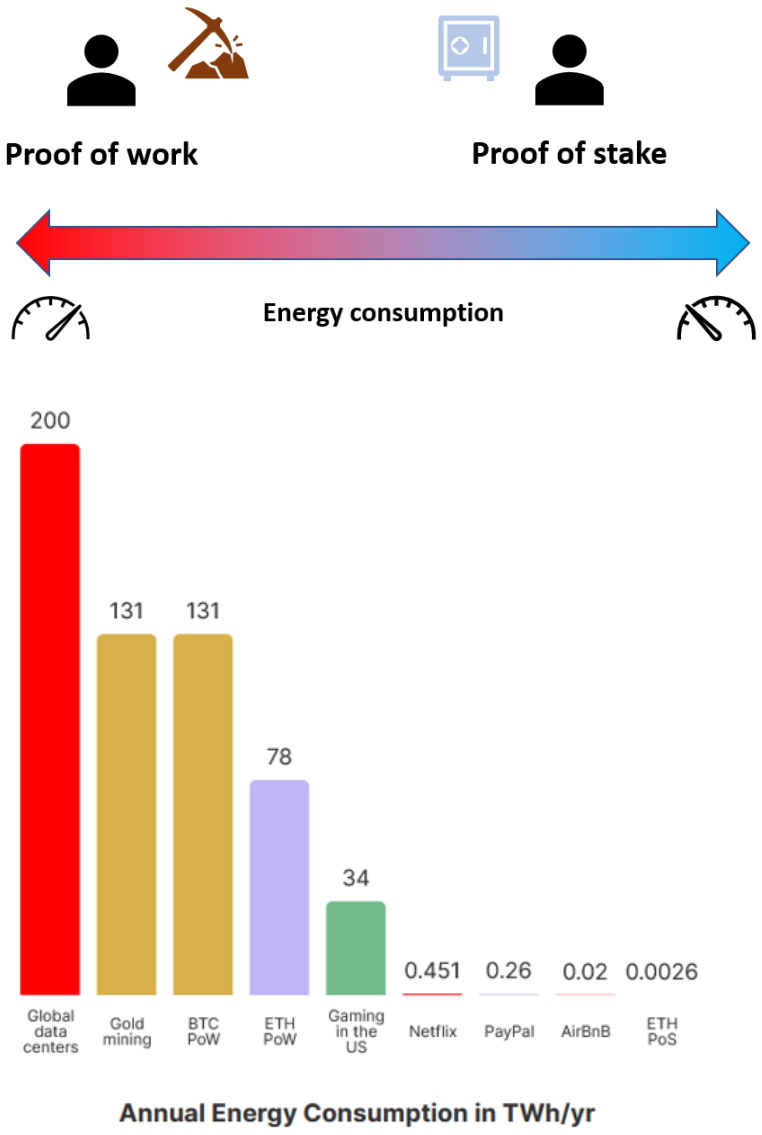

Figure 9: Illustration of energy voracity of the two most widespread systems, PoW and PoS (top) and energy consumption of different industries compared to blockchains with different consensus algorithms [ETH23a] (bottom) .

- Tezos is a relatively new (September 2018) blockchain with a lively update cycle and is widely used by European initiatives. It supports a block rate of 1 per 15 seconds but a higher transaction output (approximately 50 tps) and bears much cheaper gas than Ethereum. For comparison, the minimum stake of 8000XTZ costs over nine times less than the Ethereum minimum stake of 32ETH at the time of writing, as illustrated in Figure 10. Another interesting feature of Tezos is its self-amending chain property, granted by its on-chain governance mechanism. This

means that potential improvements are tested and voted on directly on the blockchain, which updates dynamically (without the need for forks). It also less energy-hungry than even Ethereum 2.0 [PWC21].

Hundreds of other blockchains have emerged for various purposes, including Solana [YAK18], focused on high transaction volume (it can currently support 65000tps), and Polkadot [WOO14], focused on creating a scalable ecosystem. Current trends indicate PoS as the *de facto* standard in the modern blockchain ecosystem.

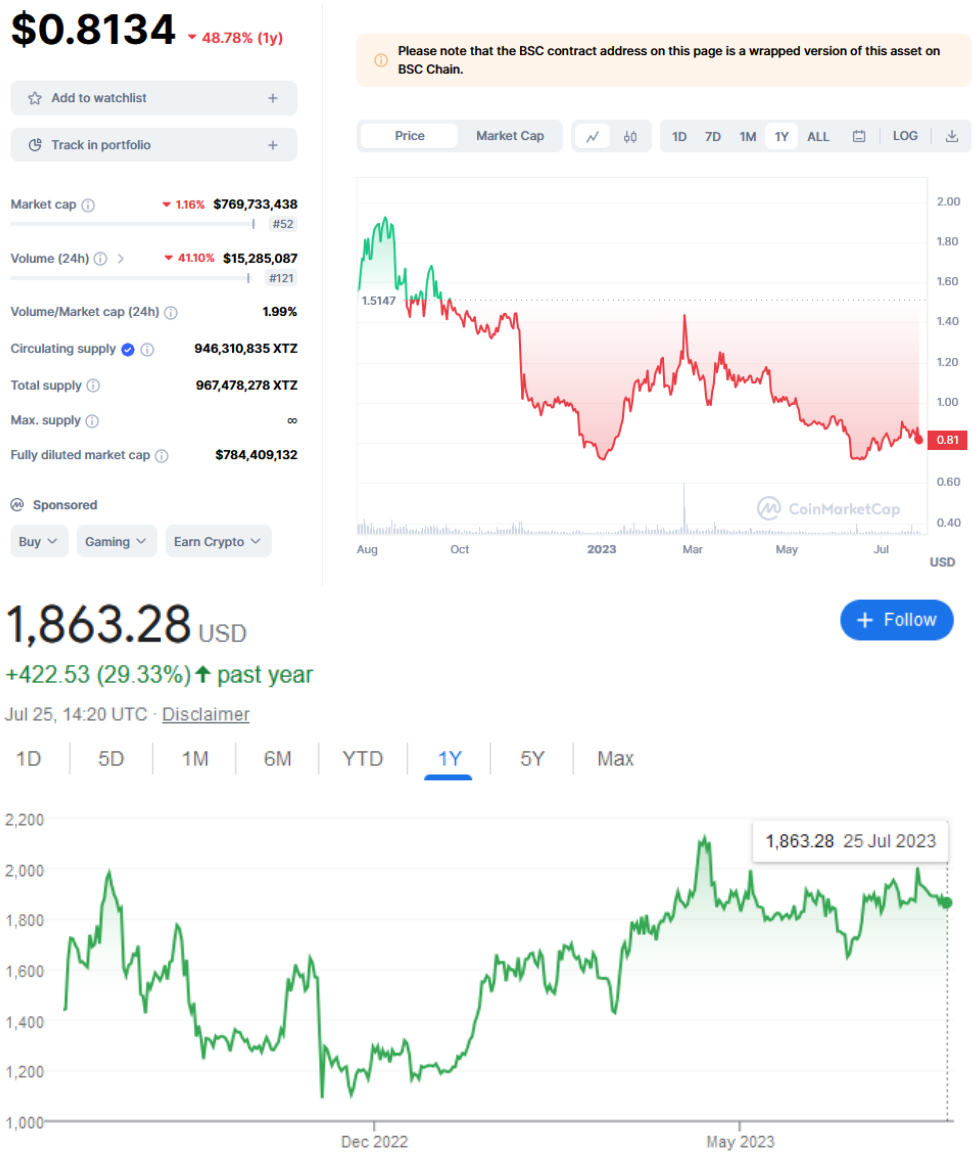

Figure 10: XTZ [CMC23] (top) and ETH [GFI1] (bottom) market valuations on July 25th, 2023 at 2:20 pm.

2.II. The applicative revolution

2.II.A. Smart Contracts

Largely considered the most impactful upgrade in blockchain technology since its creation, the emergence of application-enabled blockchains broadened the spectrum of possibilities and potential use cases. Although Bitcoin did support a form of rudimentary application support through a non-Turing-complete language called Script, its minimal scope and numerous bugs lead many to suspect this system was a late addition to the design of Bitcoin. The focus on applications began with Ethereum, a blockchain primarily built with applications in mind. In its white paper, the prior concept of *Smart Contracts* introduced in 1994 by Nick Szabo [SZA94] was launched in the web3 era. A 2021 upgrade to Bitcoin called Taproot enabled transaction-centric Smart Contracts to be executed on the eponymous blockchain.

Szabo's *Smart Contract* was defined as a computerized transaction protocol that executed the terms of a contract (*e.g.,* payments). Some early examples include Point-of-Sale Terminals and cards or digital cash protocols [SCH07]. Ethereum intended to create a protocol for building decentralized applications via an eponymous blockchain featuring a built-in Turing-complete programming language, notably supporting the creation and execution of Smart Contracts. These blockchain-aware contracts can verify conditions, execute actions using persistent variables, and run autonomously once instantiated (or *deployed*) on the blockchain. These Smart Contracts have their own accounts, including their cryptocurrency balance, contract code, and storage. These accounts are controlled by the contract code, contrarily to externally owned accounts controlled by a private key. This contract code can read and write information or create other Smart Contracts. Throughout this thesis, we shall use the blockchain definition of Smart Contract, whose etymology must be distinguished from physical contracts (to be complied with) and instead be viewed as autonomous agents [BUT14]. Smart Contracts can execute transactions (as they can write on the blockchain) or send messages to other Smart Contracts, which prompt the latter to run their code. These actions cost gas, which the Smart Contract must pay out of its balance, just like an externally owned account. Smart Contracts were initially heavily used as escrows[9] but now support a vast array of use cases, including auditing, Decentralized Finance (DeFi), anti-counterfeiting, voting, supply chain management, *etc*. A real-world example is French insurance AXA experimenting with automatically enforcing flight-delay compensations using Smart Contracts [AXA23].

Figure 11 shows a simple pseudo-code example of Smart Contracts. It holds a balance of 10 and is designed to handle a bet between two people dealing with the weather on the following day. This Smart Contract would gather information from the outside, a topic discussed in Chapter 3, Section IV.

[9] Under the blockchain framework, and for the rest of this thesis, "escrow" is understood as a third-party which holds assets until certain conditions are met [CSI23]. Escrows enable the cooperation of parties which do not trust each other.

```
contract bet{
    boolean nice_weather
    address party1= 0x001
    address party2 0x002
        if nice_weather == True
            pay party1 10
        else
            pay party2 10
}
```

Figure 11: A simple pseudo code of a Smart Contract managing a weather-related bet between party1 and party2.

Smart Contracts are often used as an enforcement for terms and conditions. Figure 12 shows an example of such a Smart Contract in a slightly higher abstraction pseudo code brought forth in [KIR17]. Both examples ignore gas costs.

```
contract terms_condition_contract {
    /* Define variables and their types */
    bool buyerNegotiationState = false
    bool sellerNegotiationState = false
    bool dealState = false
    private string dealID

    FUNCTION initialization(string msg){
        Set dealID = dealID in msg
        IF sellerFlag != 1
            sellerNegotiationState = false
            Send msg to Seller that sellerTerms has been not been accepted and to modify
        ELSE
            sellerNegotiationState = true
        ENDIF
        IF buyerFlag != 1
            buyerNegotiationState = false
            Send msg to Buyer that buyerTerms has been not been accepted and to modify
        ELSE
            buyerNegotiationState = true
        ENDIF
        IF buyerNegotiationState = true AND sellerNegotiationState = true
            Set dealState = true
            Send msg to parent_deal_contract
        ELSE
            wait
        ENDIF
    }
```

Figure 12: A terms and conditions Smart Contract pseudo code [KIR17].

For the rest of this thesis, we shall distinguish the *creation* of a Smart Contract, which we will understand as the process of writing the code and does not involve the blockchain and the *deployment* of a Smart Contract, which we will understand as the process of uploading a created Smart Contract to the blockchain, initializing its bytecode, address, *etc.* in the process.

On Ethereum, Smart Contracts are executed by a piece of software called the Ethereum Virtual Machine (EVM), which defines the rules for computing the new state of the blockchain after including a block in a deterministic fashion. The EVM cannot be described through a physical instantiation though it does exist as one entity maintained by thousands of connected computers running an Ethereum client. This client operates in a sandbox environment isolated from the computer and can only interact with the Ethereum network, ensuring isolation from tampering. Various modern blockchains are now EVM-compatible, *i.e.,* capable of running the EVM and executing Ethereum Smart Contracts.

Ethereum Smart Contract code is treated in an EVM code, a low-level, stack-based bytecode language (which resembles assembler output). Yet, developers do not write code in this language directly but in Solidity, a high abstraction object-oriented programming language inspired by C++, JavaScript, and Python and targeted at the EVM. It notably supports inheritance, libraries, and user-defined types, as modern developers have gotten used to. Thorough examples of Solidity Smart Contracts will be given in our implementation chapter (cf. Chapter 5).

Since Ethereum, almost every blockchain has been built to support applicative needs. Each made decisions regarding their protocol's philosophy and programming languages' features. For comparison, Tezos uses a specifically designed programming language called Michelson for its Smart Contracts. Contrarily to Solidity, Michelson produces an easily understood bytecode and supports formal verification of program conformance to a given set of specifications (a systematic step-up from unit tests). This leaves Smart Contracts to deal with relatively lightweight transaction logic to manage. These features are handy for financial or escrow contracts which involve significant financial resources. Tezos developers write their code using higher abstraction languages such as LIGO [LIG23] or SmartPy [SPY23a]. We will use the latter for our Tezos implementations (cf. Chapter 5), as it resembles standard Python. Interestingly, SmartPy constitutes meta-programming, where the code written is used to generate a Smart Contract [SPY23b].

Blockchain Basic : A Smart Contract is an automated piece of code deployed on a blockchain. Smart Contracts are programmed via specialized software languages with unique features and limitations.

2.II.B. Blockchain standards

The International Standard Organization's (ISO) involvement in blockchains was put into full effect with the creation of ISO Technical Committee 307, Blockchain and Distributed Ledger Technologies [ISO16b], which hosted working groups focused on subjects going from blockchain terminology to governance for industry and government usage. These working groups were diverse in the countries that participated and connected with other

major organizations such as the European Commission, the International Telecommunication Union (ITU) [ITU1], and even the Society for Worldwide Interbank Financial Telecommunication (SWIFT)[SWI23]. Other national and international standard-defining organizations, such as the National Institute for Standards and Technology (NIST) [NST23] or the Institute of Electrical and Electronics Engineers (IEEE) [I3E23], followed suit, creating subdivisions and working groups. Moreover, blockchain-specific organizations such as the International Association for Trusted Blockchain Applications (INATBA)[INATBA] saw the light of day. All these institutions provide policies, frameworks, industry applications, and myriad other theoretical contributions to the field and sometimes fund blockchain-backed projects serving not only industries but also consortiums, states, local governments, *etc*.

Yet, in blockchain jargon, the word "standard" does not automatically refer to documents and formats issued by international standard organizations but to various *de facto* applicative standards whose terminology we will simplify by also referring to them as standards, as is the norm in the community. These community-driven standards are much more technical and pragmatic to day-to-day blockchain users.

For instance, Ethereum relies on Ethereum Request for Comments (ERC) and Ethereum Improvement Proposals (EIP) standards, which anyone can create, granted they explain their idea clearly and foster enough community support [EIP23]. ERCs focus on the rules and specifications related to *Smart Contracts* and *tokens* (cf. Section II.C), e.g., API streamlines (ERC4626). EIPs target the overarching protocol, e.g., changes in the transaction fee model (EIP1559). The latter typically follow EIP1 guidelines, must provide technical specifications and garner a consensus, document alternative opinions, and are usually advanced by high-level application or protocol developers. EIP editors (as defined by EIP5069) then decide whether a proposal should become an EIP and suggest potential improvements. Once accepted, ERCs are adopted by developers, while EIPs are implemented in future versions of the Ethereum protocol.

As we alluded to earlier in this chapter, Tezos functions differently. The basis of the Tezos Improvement Process (TZIP) is familiar, as anyone can submit a document that presents a new feature or specification supporting the formal protocol governance process to address to the community [TZP23]. It contains the rationale and the technical details of the proposal and should uphold answers to common questions and alternative options. Once voted in by the community (this process is on-chain), the proposal is dynamically implemented fashion according to Tezos' off-chain governance.

Blockchain Basic: Despite *de jure* standardization initiatives, blockchain applicative environments make heavy use of *de facto* community-driven standards.

2.II.C. Tokens

Some of the most notable and widely used application standards define the formats of *tokens*, the blockchain representation of assets. The cryptocurrency supported by a blockchain is a token often referred to as a *protocol* token, *native* token, or *transactional* token. As explained earlier in this chapter, these tokens are essentially the local currency of the environment and are used to compensate miners in PoW or to send value to other accounts. Examples of transactional tokens include ether for Ethereum, which is often symbolized by ETH, or tez for Tezos, which is often symbolized by XTZ.

More interestingly, users can also create tokens to represent digital assets such as a piece of art, a diploma, proof of ownership of a physical asset, or even a ticket to an event. These asset formats are defined by standards that typically divide them into two categories: *Fungible Tokens* (FTs) and *Non-Fungible Tokens* (NFTs). FTs can be split and are interchangeable (*e.g.*, a specific token is worth as much as any other, just like a dollar), whereas NFTs are unique and cannot be split. *Token Contracts*, a subset of Smart Contracts, are deployed and define the functionalities and manage the supply of tokens they represent. Most notably, they record a mapping of account balances: an account owning a token means its addresses' balance is nonzero in the Token Contract. Similarly, sending tokens means calling the appropriate Token Contract's function that affects these balances.

Blockchain Basic: Tokens are the blockchain representation of digital assets and are formatted by applicative standards.

Some noteworthy FT standards include Ethereum's ERC20 [SMI23a] or Tezos' TZIP7 FA1.2 [TEZ23a]. At their core, these standards include a ledger that map token balances and an API for transfer and approval operations. Figure 13 shows an interface of the ERC20 standard, omitting optional functions, and a simple illustration of the actions permitted by this interface. This illustration omits the distinction between tokens and Token Contracts to illustrate functionalities.

```
FUNCTIONS
totalSupply()
balanceOf(account)
transfer(recipient, amount)
allowance(owner, spender)
approve(spender, amount)
transferFrom(sender, recipient, amount)
increaseAllowance(spender, addedValue)
decreaseAllowance(spender, subtractedValue)
_transfer(sender, recipient, amount)
_mint(account, amount)
_burn(account, amount)
_approve(owner, spender, amount)
_burnFrom(account, amount)
```

```
EVENTS
Transfer(from, to, value)
Approval(owner, spender, value)
```

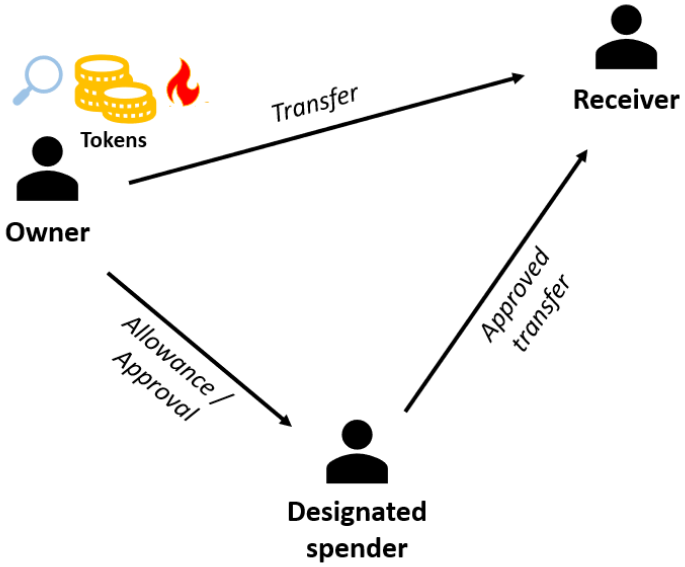

Figure 13: ERC20 Ethereum fungible token standard functions and events [OPZ23b] (left) followed by an illustration of the roles and actions defined by the standard (right).

Briefly, the first two functions deal with supply and return integers, the two following deal with giving other addresses spending rights over tokens that they do not own, `transferFrom()` enables to send tokens to another account (external or Smart Contract), and the following two allow to adjust allowances given using the `allowance` function. All these are public, which does not mean everyone has every right, but everyone is free to call these functions. The rest of the functions are internal, meaning they can only be called from within the Smart Contract. Vocabulary-wise, *minting* a token means creating an instance of a token, and *burning* means deleting a token. The Transfer and Approval events are emitted when certain previous functions are successfully called and can be caught to launch other events by other pieces of code *on* and *off* of the blockchain.

Noteworthy NFT standards include Ethereum's ERC721 or Tezos' TZIP12 FA2. Multi-asset standards, *e.g.,* ERC1155 or FA2, emerged to streamline token variety and eliminate redundancy by enabling the representation of any number of FTs and NFTs simultaneously without having to deploy multiple contracts. These standards API resemble the one shown hereabove. NFTs are sometimes used as representations of digital art and sold as such, powering a massive market we shall analyze in Chapter 3.

Blockchain Basic: Token Contracts are Smart Contracts that manage tokens and track account balances.

2.III. Historic weaknesses

2.III.A. Infrastructure and consensus issues

The primary issue blockchain skeptics justly raise is the lack of understanding of the infrastructure and its advantages and drawbacks with regards to legacy solutions. This issue has resulted in the use of blockchains because of their mainstream appeal as opposed to technological requirements. Blockchains have repeatedly been invoked where simple databases or intranets would have flourished without a clear view of the benefits of the DLT component [KAZ21]. The reality of blockchain shortcomings hit these initiatives later or sooner, *e.g.,* as 92% of 86,000 blockchain projects tracked by Deloitte in 2017 were abandoned before the end of the year [TRU17]. These observations were sometimes summarized in the saying "Blockchain is a solution looking for a problem" [BRO18]. On the contrary, some who have garnered an extensive understanding of the scope of the technology and its original philosophy have brought a pejorative meaning to "zero trust": many blockchain applications cannot be trusted and are simply scams.

Moreover, this misunderstanding can also occur in the specifics of a blockchain once it has been decided as a solution. Specifically, some point out that it is the combination of security features that makes the original blockchain design robust, and that no single feature can carry that guarantee by itself, and that many of the implementations sprouting in industry applications neglect this fact. In an incendiary article [HAM16b], cyber security expert Nikolai Hampton claims that "many in-house blockchain solutions will be nothing more than cumbersome databases" and that "without a clear security model, proprietary blockchains should be eyed with suspicion." This claim is backed by the observation that "There is also no need for a '51 percent' attack on a private blockchain, as the private blockchain (most likely) already controls 100 percent of all block creation resources. If you could attack or damage the blockchain creation tools on a private corporate server, you could effectively control 100 percent of their network and alter transactions however you wished". While private PoA blockchains have developed schemes with BFT featuring careful role attributions to ensure high decentralization, private blockchain implementations have scarcely put this theory into practice, usually centralizing authority, going against the very philosophy of blockchains.

Even with the proper understanding and implementation, blockchains suffer from significant scalability issues by very design, as every transaction needs to be processed by every node in the network, *e.g.,* the Bitcoin blockchain, which is currently about 15GB, grows approximately 1MB per hour. If Bitcoin were to support all Visa payments (around 2000 per second), it would grow 1MB every three seconds, making it 1GB heavier per hour, 1YB per year. Despite its advantages in design (namely the state recording rather than history recording), Ethereum is currently and will continue to face this issue bolstered by the presence of applications on top of currency exchanges [VER18] [BUT14]. To anchor our point in a more realistic example, the Cryptokitties [CRY23] blockchain-based decentralized game caused a sixfold increase in pending Ethereum transactions in 2017, slowing the network noticeably [EVA19].

Fortunately, the above issues all have simple solutions. Applications stemming from a lack of understanding will likely not stand the test of time. The drawbacks of the technology

(processing times, energy consumption, *etc*.) will eventually catch up with unfounded use cases and leave blockchains to problems appropriately tackled by the technology. This time will also contribute to the overall understanding of the ins and outs of blockchains, which should increase the standardization of the environment and decrease the general vulnerability to scams. Regarding fundamental blockchain issues, historical actors have remained up to date with potential issues and rolled minor and major changes alike (*e.g.,* Ethereum 2.0). Younger actors, on their side, have had the opportunity to learn from their predecessors, which can be notably seen in the multitude of solutions brought forth concerning scalability issues in new infrastructures.

2.III.B. Application exploitations

Despite everything they bring to the environment, Smart Contracts are not without their limitations. Their role often leads to comparisons with web2 programming, which they cannot match on several points.

First, Smart Contract language (regardless of blockchains) performances and functionalities are not up to par with legacy programming. As of writing, these languages lack the variety of general-purpose libraries that developers have gotten used to. Also, Smart Contracts must be developed with a focus on efficiency and the constant worry of gas expenditure, not as second-thought optimization [ZHA20a]. Combining these limitations that the industry has overcome in web2 but still exist in web3 and the learning curve of unfamiliar concepts, Smart Contract development has seen some setbacks in its widespread adoption. 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.

Key Issue: Smart Contracts should not be seen as a transposition of web2 software as they cannot match web2 software one-to-one.

Although these high abstraction limitations are far from negligible, the biggest adoption issue with Smart Contracts comes from an integral part of their design: they are immutable. Although self-executing, non-negotiable, public pieces of code enable a new paradigm, said paradigm is unwavering and sometimes challenging to navigate. This means that Smart Contracts cannot react to unanticipated events and are also highly prone to human error. Indeed, a typo or the simple overshadowing of an aspect of the code and a Smart Contract could be unable to perform its intended purpose and lock up funds or be exploited by malicious actors. For instance, forgetting to include an allowlist on a function meant to send funds back to an owner means that anyone can empty the contract of its cryptocurrency. A recursive function with no stop condition will eventually drain the contract's balance with gas costs. In short, a Smart Contract cannot be developed further

once deployed and must be, in a sense, "perfect." As the saying goes in the blockchain community: "Code is law" [LES00].

Then, a significant drawback of Smart Contracts lies in their inability to seamlessly retrieve off-chain data. Their design ties them to the status of the blockchain, while real-world applications would want to implement Smart Contracts in larger systems and have them communicate with other technologies or APIs. Bridges between Smart Contracts and external data sources called *oracles* have emerged to compensate for this shortcoming. Specifically, Decentralized Oracle Networks (DONs) enable the creation of hybrid Smart Contracts rooted in off-chain infrastructures where they can retrieve data. These oracles can get data to Smart Contracts (Input oracles), get Smart Contract data to off-chain systems (Output Oracles), enable interoperability between blockchains (cross-chain oracles), or enable impractical operations to be run off-chain (compute-enabled oracles) (cf. Chapter 3, Section III).

Key Issue: Smart Contract development and deployment is tedious and prone to costly mistakes. Smart Contracts also struggle to seamlessly exchange data with off-chain solutions.

Once again, these limitations are not prohibitive to the existence of Smart Contracts. Their user-friendliness has significantly improved over the years, and regular updates continue to emerge and improve usability on every active application-driven blockchain. Similarly, solutions to their technical limitations continue appearing as parts of their design or as complementary solutions. Moreover, although costly mistakes were common in the early days of Smart Contracts, mishaps have become much rarer in a world where automatic reviews, in-depth testing, an active and knowledgeable community, and trusted shared applicative standards are all dedicated to making Smart Contract use as safe as possible. The community has also observed a shift in approach, understanding that Smart Contracts are not replacements for all programs and are specialized to operate within certain boundaries.

2.III.C. Blockchain hacks and pseudo-hacks

Although this thesis does not directly deal with the security of the network, the robustness of the protocol, nor the regulatory aspect of blockchains, a general understanding of the historic faults of the system and successful attacks perpetrated on it and its complimentary tools is essential to maintain a proper perspective on the subject. As such, the following will mention and briefly explain the most notable events in the history of blockchain attacks and their relationship with regulatory institutions.

Blockchains were built and are known as robust and secure infrastructures, but they are not immune from technical or social attacks. Decentralized networks have always been vulnerable to the "double spend attack," where malicious users manage to simultaneously spend and keep assets by altering some records. Blockchains solve this issue thanks to the

broadcast of timestamped transactions. As Satoshi Nakamoto mentions in the Bitcoin white paper, ‘‘peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions.’’ The only way to bypass this system is to alter the record, which could be done by taking control of the network through a “51% attack”, which aims at gathering enough computational power to overpower honest validating nodes and essentially “re-write history”.

A 51% attack consists of a miner accumulating enough hash rate to inject a chain of blocks into the network. This chain of blocks is generally mined privately. It is then pushed on the network with significant computational backing to invalidate previous valid transactions that spent the attacker’s assets, effectively recovering them after they have been spent. It is essential to understand that this is not a hack *per se* but an exploitation of the consensus mechanism. General user’s funds are safe and are not devalued by the creation of new assets; no private keys are exposed, but a specific receiver is led to believe they received assets. The attacker can subsequently benefit from a service they pay for before making their transaction disappear in history, recovering their assets.

As mentioned in this chapter, this attack is practically impossible on modern Bitcoin due to the amount of power it would require. Moreover, this strategy would only provide a marginal advantage, as it would simply improve the probability of controlling the next block. [BUT13] shows that a 25% “economic” attack would be sufficient in the context of selfish mining, forming a coalition of miners acting in their self-interest. Still, the author (referencing early Bitcoin forum posts as predecessors), points out that this attack would need to be announced in advance, making it *quasi-null*. The paper also shows that smaller groups of miners can affect the system with similar strategies. Moreover, the participants in the network are strongly incentivized to act in the best interest of the *status quo*, not only for ideological reasons but simply because it supports their income source. [EYA18], a seminal paper on the subject, goes in-depth on mining shortcomings and game-theory considerations. It established one of the first strong propositions: “Bitcoin’s network security is slightly less infallible than we at first might think it is” [BUT13].

51% attacks have occurred on early Bitcoin and smaller blockchains, notably Bitcoin Gold (a hard fork of bitcoin) in 2018 and 2020 and on ZenCash in 2018. Earlier scares of mining pools becoming too big did occur in 2014 (notably with the GHash.io pool). Yet, not all these attacks were able to be confirmed as profitable, and the respective blockchains answered by raising the amount of confirmations transactions required, introducing penalties for delayed block reporting (which enables attackers to create long chains which can be selected), limiting the capabilities of large mining pools, and introduced redundant pointers to highlight parallel blocks [ZEN18]. Other consensus mechanisms have approached the problem proactively. For instance, LPoS punishes the creation of same-level blocks (double baking) by slashing the baker’s stake and rewarding correct accusers.

The most significant and impactful blockchain attack in blockchain history is undeniably the Decentralized Autonomous Organization hack of 2016. DAOs [LIU21b] are blockchain cooperatives run by code (“code is law”) that replace centralized management structures and are particularly popular in DeFi. An Ethereum DAO was launched in 2016 and sold voting rights tokens in addition to collectivizing business funds to distribute investments

and dividends. It raised 150 million USD and became of the largest crowdfunding campaigns in history. Yet, significant vulnerabilities in the DAOs Smart Contracts were found and exploited, siphoning the funds away. This event prompted the proposal of a fork (first soft, blocklisting the attacker, then hard, solving a major bug), which split the community (not without pressure from the attacker, claiming his actions were legal and that he would bribe miners not to comply with a fork). This event crystallized the complex positioning of blockchains at the center of technical and moral considerations. The hard fork was eventually executed and reverted Ethereum to its state before the attack, redistributing the stolen funds to the victim. This sparked major controversy as it questioned the immutability and the resistance to central authority blockchains such as Ethereum proudly display. Those who refused the fork remained on what is now known as Ethereum classic, which is still operational today.

Taking a step back, blockchains proved themselves to be very secure networks, as Bitcoin and Ethereum, the two biggest infrastructures, have never been hacked. Yet, this does not mean much when other facets (*e.g.,* consensus exploitation) present weaker links to target or security protocols are not adequately upheld (*e.g.,* the infamous 2014 NXT hack [MEH19]). In addition, almost all users rely on third-party software for their blockchain-roaming activities (wallet providers, cryptocurrency exchanges, cross-chain bridges, *etc.*), software that they have not (or cannot) verify, which has repeatedly shown it can be hacked (*e.g.,* Ronin network [HEN23], Wormhole bridge [SCH23]). Yet, the most common way for hackers to operate in the blockchain environment is through social engineering and phishing, reinforcing the need for everyday security practices, cold wallets, and careful private key management as the best ways to remain safe.

Finally, blockchains are still in an uncertain position regarding general regulations. Their relative novelty and high disruption make laws hard to craft, while their governance and infrastructure make it hard even to consider which framework to legislate them under. Blockchain transfers ignore borders, statuses, and financial regulations and make personal information retrieval impossible for government oversight and taxation. Moreover, blockchains have proved to be very useful for criminal activities and illicit markets because of the lack of traceability and anonymity. Further services have even spawned in blockchains to further disguise the origin of funds, *e.g.,* tumblers that mix funds from different sources before redistributing them to obscure trails. Such a service, Tornado Cash, was blocklisted by the US Treasury in 2022 [TRE22], which highlighted the regulatory tension in the space.

These observations led to the emergence of specialized blockchain tracking services, which have pushed shady operations to less regulated blockchains. This game of cat and mouse is familiar to hackers and security experts, but blockchain adds an underlying debate on data privacy and central authority oversight.

Blockchain Basic: The biggest blockchains have never been hacked, although third party solutions in the space have been (by virtue of the *weakest link* principle). Misplaced trust and lost private keys are bigger threats than sophisticated attacks on the protocol.

2.IV. Summary

Logically, a blockchain consists of the following layers: infrastructure, networking, consensus, data, and application, as illustrated in Figure 14.

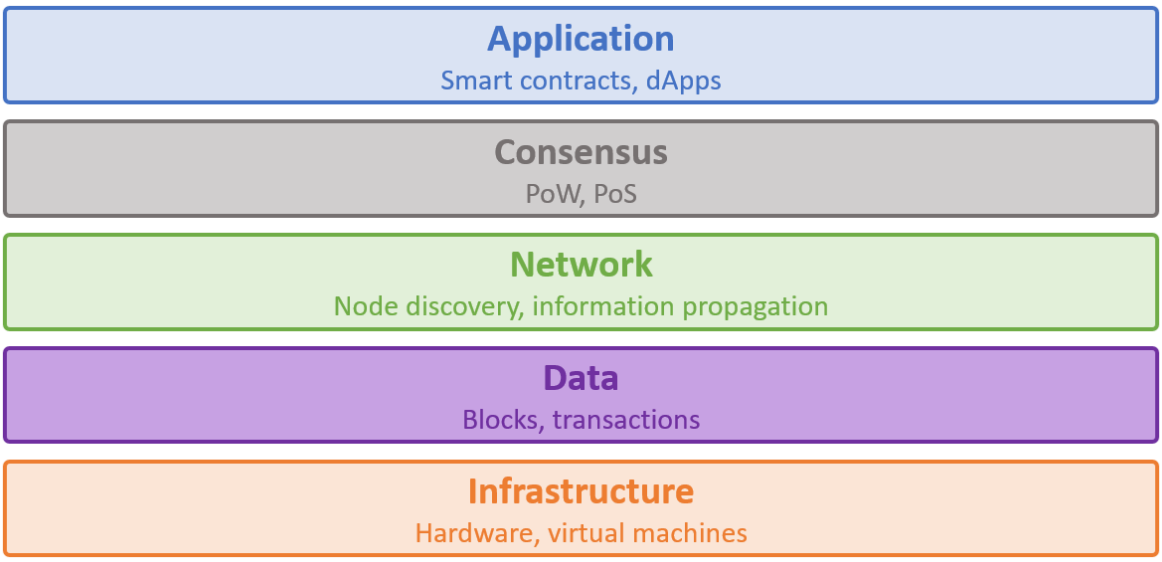

Figure 14: Applicative blockchain technological stack.

Users can exchange information and assets in local cryptocurrencies through anonymous accounts. Applicative blockchains now allow third-party software development, which relies on two fundamental pillars: Smart Contracts, automatized pieces of code living on blockchains, and tokens, the representation of assets on blockchains. Both have wide arrays of use cases driven by community *de facto* applicative standards and regulatory bodies.

Although attacks on the infrastructure and third parties have shown possible, the network is highly secure, especially when widely documented best practices are applied. Besides, blockchain regulatory statuses regarding off-chain authorities are still uncertain, which promises lively debates in the near and distant future.

Chapter 3. State-of-the-art

This chapter will be dedicated to a thorough investigation of the current state-of-the-art of the field beginning with an evaluation of Smart Contracts, token distribution, and token protection. Then, an analysis of current inter, intra, and extra-blockchain interoperability approaches will enable us to identify current gaps in the literature we will address in the rest of this thesis. Finally, we introduce practical concepts from off-chain vertical fields that will be used in conjunction with blockchain in the elaboration of our solutions.

3.I. Smart Contracts and Decentralized Applications

"Smart Contract" has become a selling term in itself over the past years. The primary use of Smart Contracts, namely the capacity to enforce agreements between parties without the involvement of a trusted third party, allowed them to gain massive traction in the DeFi and notarization fields, as summarized in [DHI22]. The most significant restriction to new applications was the absence of proper interfacing, which left only Smart Contract-comfortable users to interact with them. This glass ceiling was finally shattered by the concept of dApps, the web3 evolution of now familiar applications. From a user perspective, dApps do not differ from their web2 counterparts, but their backend is powered by decentralized networks and web3 paradigms (not limited to blockchains, unlike DeFi). These dApps emerged on the Ethereum blockchain thanks to the Turing-completeness of Solidity and further prospered thanks to the existence of the Token standards for represented assets (ERC20 emerged in 2015, ERC721 only being proposed in 2018, far from being the first NFT). Their main advantage remains in the line of blockchain asset exchanges and lies in the absence of a central authority. Dapp interfacing in illustrated in Figure 15.

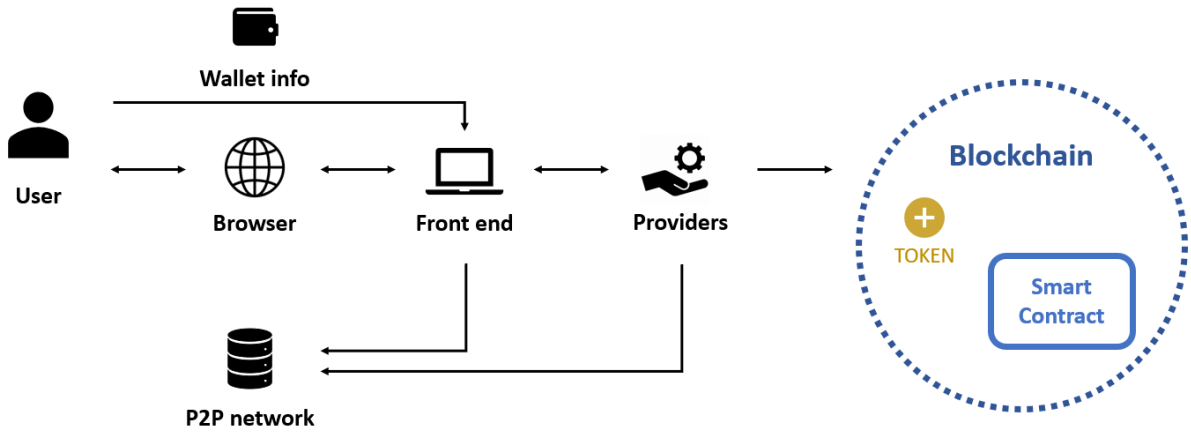

Figure 15: Technical interfacing of decentralized applications, which rely on a web2 front-end and a web3 back-end.

The first notable example of dApp was founded in 2014, called MakerDao [MAK23], and existed within the context of DeFi. Its role was to create a cryptocurrency, namely DAI, whose value remains tied to the value of the USD thanks to collateral (digital) assets. Its main goal is to undermine the volatility associated with DeFi while retaining the advantages of decentralized backends. DeFi-lead dApps continued to prosper, and we notably saw the emergence of collateralized loan platforms and *swaps*, where users can exchange cryptocurrencies without passing by fiat currencies. With these tools, users could use cryptocurrencies without selling them for fiat to buy new ones, making DeFi independent of trust-requiring (and not zero-trust) assets. New use cases for dApps quickly followed and guided the evolution of applicative blockchains. For instance, the Cryptokitties network slow down led to Ethereum increasing its gas limits and TPS

capacity and paved the way for modern NFT exchanges [CRY23] [EVA19]. Other than gaming [MIN19], some notable fields for dApps include healthcare [ANG17], gambling [GAI17], or social media [LI19].

These use cases highlight the diversity of outlooks and embody the main strengths and weaknesses of the dApp archetype. DApps do not run entirely on decentralized networks but store data and relevant assets on blockchains. As such, no central entity possesses user account information and can affect the application singlehandedly. Furthermore, user actions are easily seen and can be validated. Various approaches with different levels of decentralization exist, some applications relying on elected moderators or developers while others rely on governance tokens in a DAO fashion [DAP23a].

The main limitations of dApps at the time of writing and include:

- **Computational limitations:** Current web3 infrastructures cannot support many on-chain features. This has notably limited the ability of real-time applications [MIK17].
- **Scalability:** Linked with the above, the growth of modern approaches is severely limited by transaction speed and load [SCH17].
- **Technical knowledge requirements:** DApps may require users to manipulate cryptocurrency wallets, token exchanges, *etc*., which are a significant barrier to entry [ZOU19].
- **Interoperability:** Blockchain dependency, exclusive wallets, and non-exchangeable assets are the form of lack of interoperability takes for dApps [KHA21b].

Putting dApps in perspective with the rest of the software field, it is essential to remember that initial innovations emerged with little regard to non-web3 regulations, such as taxation, contributing to the flux of opportunistic users. Moreover, connecting public, immutable peer-to-peer backed code with historically very vulnerable web2 software is not without risks. The *weakest link* security principle [ARC03] applies and states that the resilience of a software architecture mostly depends on protecting its most vulnerable components. The concept is represented in Figure 16. The robustness of web3 paradigms is irrelevant if an attacker can easily access private keys or react to broadcasted events. The industry caught up with this fact and now relies on automated testing and full audits by specialized businesses [HE20].

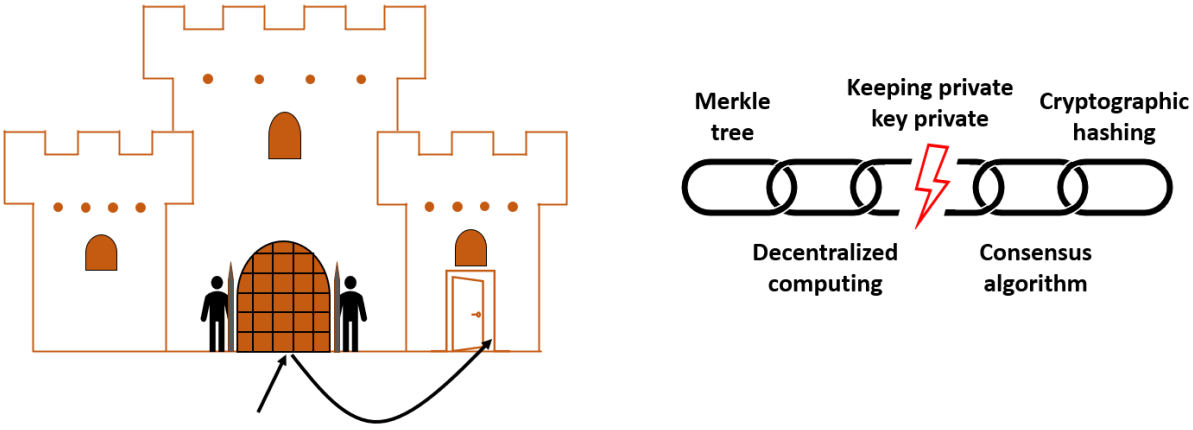

Figure 16: Medieval (left) and web3 (right) illustrations of the weakest link security principle.

This fact does not prevent Smart Contract limitations from being investigated for technological and adoption shortcomings. Smart Contract security, programming language limitations, and performance problems are specifically identified as significant challenges developers face when tackling applicative blockchain environments [ZOU19]. In practice, these issues materialize in the risk and consequent fear of unmodifiable mistakes happening. This fact is bolstered by Smart Contracts being public and targetable at any time. Moreover, quality of life features offered by blockchain development are far from the level of comfort developers have been accustomed to. Some examples include the constrained number of local variables, the absence of general-purpose libraries, and the absence of systematic support. As a result, even those who took the leap must keep Smart Contracts simple and intuitive as a matter of best practice and avoid them becoming prohibitively expensive or significant liabilities.

Critical Limitation: Smart Contracts are significantly limited in their computational capacities, scalability, and quality of life features.

Given the importance of Smart Contract design and the irreversible consequences mistakes can have, automatic and semi-automatic Smart Contract generation has emerged several times in the literature. This approach ensures generated Smart Contracts follow a given pattern which exemplify established business workflows. For instance, [ZUP20] proposed the generation of Smart Contract templates using Petri Nets [PET77] which can be further deployed with little effort. A more systematic approach based on human-readable contracts to be translated into Smart Contracts was taken by [FRA16] and [TAT19]. The former specifies Smart Contract components that correspond to real-world operations through a domain-specific language, while the latter is based on Controlled Natural Language (CNL) mapped to Smart Contract models. The subject is now an active research topic in the world of standards, most notably in ISO/IEC 23093 and ISO/IEC 21000, respectively in the fields of IoMT and media.

The current landscape of dApps is filled with new initiatives from a mixture of businesses, agencies, and consortiums. The most proven uses, past DeFi dApps, are in supply chain management [COL19] and identity verification [JAC16], which benefit the most from the transparency and inherent trust provided by the blockchain environment. The activity of start-up businesses and the scientific literature in the field suggests that opportunities are still being investigated for various industries and use cases [HEW21].

Critical Limitation: The necessary resources to properly create and deploy Smart Contracts makes them sometimes impractical in use cases featuring highly repeatable workflows.

3.II. Tokens and their distribution

As explained at a technical level in Chapter 1 and Chapter 2, the second central notion of applicative blockchains is the token, *i.e.,* the blockchain representation of digital assets, which are formalized by widely accepted *de facto* blockchain-specific application-level working standards [ANJ17]. These assets take several forms and can be owned and traded by user accounts and Smart Contracts. They can represent rights in IPR management schemes or virtual assets in a myriad of cases (*e.g.,* decentralized games). Yet, the fact that tokens (and NFTs in particular) made for hundreds of headlines [WAN21a] over the past years and that even people with no knowledge of the web3 space are regularly familiar with the space can be attributed to one specific use: NFTs are often used as representations of digital art. This form of web3 media gained a lot of traction by promising to put artists directly in control of their IP and by claiming qualities such as verifiable authenticity and transparency.

Before we delve further into their distribution and market, we first detail the anatomy of NTs, shown in Figure 17. A digital art NFT does not contain the data of the underlying asset it represents but:

- A digital token containing metadata that often points to off-chain storages (usually InterPlanetary File System, or IPFS) where the artwork is stored,
- A license that establishes exploitation rules.

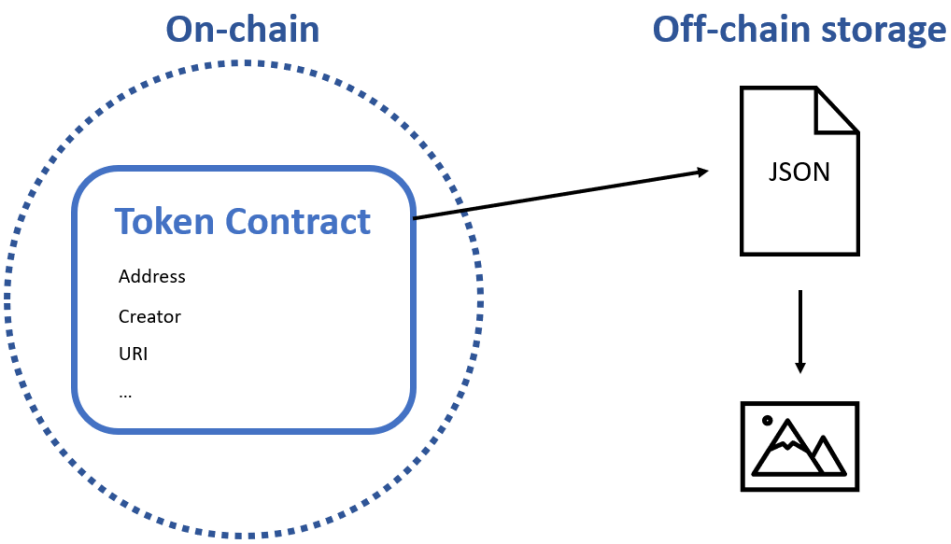

Figure 17: Overview of the anatomy of tokens and their link with off-chain storage.

This means that token IP is much more centralized than one might think, and buyers are severely restricted in what they can do with the image represented by their NFT. Very few NFT collection provides IP ownership, and a sizable portion only deliver personal use or creative commons licenses.

- Personal use license: "*an intellectual property license that does not allow any type of commercial use or exploitation of such intellectual property; specifically, a Commercial License delivers a limited, non-exclusive non-sublicensable, non-transferable, non-assignable license to use or consume such Content for internal or personal purposes only, without any authorization to publicly perform, display publish, make derivates of, or financially exploit such Content in any manner whatsoever.*" [LAW23]
- Creative Commons license (CC BY): "*This license lets others distribute, remix, adapt, and build upon your work, even commercially, as long as they credit you for the original creation. This is the most accommodating of licenses offered. Recommended for maximum dissemination and use of licensed materials.*" [CCO23]

Key Issue: Tokens do not always carry the IPRs owners assume and sellers advertise.

Consequently, all the NFTs distributed with such licenses (a majority, according to [THO22]) do not convey IP ownership of the underlying content. This becomes an even bigger problem when significant actors in the field falsely claim "you own the IP" on their websites. Only very scarce collections even attempt to transfer the IP to token buyers (the World of Women collection was the only one in the top 25 in 2022), and the most common licenses render token ownership useless, as purchasers have the same rights as non-purchasers on the underlying media. All of this hurts the credibility of tokens for specialized and non-specialized publics. Still, more importantly, it does not support the expansive vision of web3, which largely relies on decentralized assets.

But this goes even further, as the ownership of a link and not the data at that link means that data can be modified without breaching the purchase agreement. This fact was showcased in the many NFT projects that used mass marketing to generate hefty investments before disappearing and replacing the underlying data with the image of a rug, a play on the "rug pull" strategy[10] employed by the fraudsters. To spark conversation about the topic, Geoffrey Huntley created the NFT Bay [NFT23], an archive containing 18TB of data represented by NFTs and 12GB of the links to IPFS storages contained in the original NFTs. Going back to what NFTs are, the first was bait and completely legal (assuming none of the tokens had extremely strict IP Licenses), while the latter targets and distributes the link that was purchased and is owned by other people and is subsequently legally ambiguous at the very least. The former was called "the heist of the century" by dozens of outlets, while the second was ignored before the creator came forward himself. An interview with Huntley can be found at [YTB22], where he highlights his message, "What are people actually purchasing?".

With the understanding of NFTs as the representation for digital art, one can move on to the notion of *marketplaces*: or decentralized applications connecting NFT buyers and sellers. Given the growth in popularity of NFTs, a wide array of these marketplaces emerged to acquire a portion of a market [DAP23b] worth 343 billion USD in the second quarter of 2021 alone [CON21]. In addition to buying and selling NFTs, marketplaces now sometimes allow to *mint* (or create) new assets. While it is true that users could mint their own NFT and directly send them to other addresses, a vast majority of the trading volume happens via marketplaces. This is the case because these platforms bring user-friendliness and allow users to advertise their creations, a particularly arduous task for an individual with no prior following in an environment designed for anonymity. While they started as platforms allowing people to buy assets for cryptocurrency, they now provide a wide array of features, regulations, fees, and rules. Figure 18 shows the evolution of the Ethereum NFT marketplace monthly volume, peaking in February 2023 with over 1.7 billion USD [BLO23].

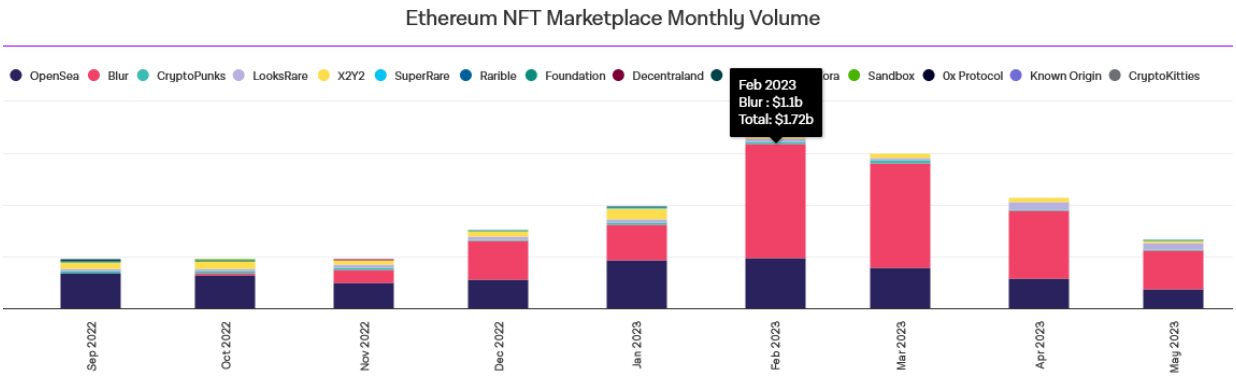

Figure 18: Ethereum Non-Fungible Token marketplace monthly volumes and market shares between September 2022 and May 2023 [BLO23], peaking at 1.7 billion USD in February 2023.

[10] A "rug pull" designates a sudden revelation that completely contradicts the assumptions one has been led to believe.

Blockchain Basic: Marketplaces act as delegated trusted parties to handle the exchange of Non-Fungible Tokens.

Key Issue: Token distribution has become intrinsically linked to digital art and third-party exchanges.

The only common aspect of all marketplaces is their *modus operandi*, which typically resembles some variation of a *Swap Contract*, which functions as illustrated in Figure 19. First, the platform initializes a Smart Contract with the desired set of rules. It contains a mapping of tokens to their respective owners and prices, illustrated in Figure 19's central Swap Contract. Then, the Smart Contract receives and holds the tokens until they are sold or reclaimed by their original owners, as illustrated by Step 2 Figure 19. A buyer who wants to purchase a token on the marketplace can then use the Smart Contract's `buy()` function [TEZ23a], [VOG15], [ENT18], Step 3 Figure 19. The Smart Contract ensures proper payment of the appropriate parties and keeps a portion to cover the marketplace's fee (Steps 3.B., 3.C. Figure 19) before sending the desired token to the buyer (Step 3.D. Figure 19). A fully implemented example can be found at [GTB22a].

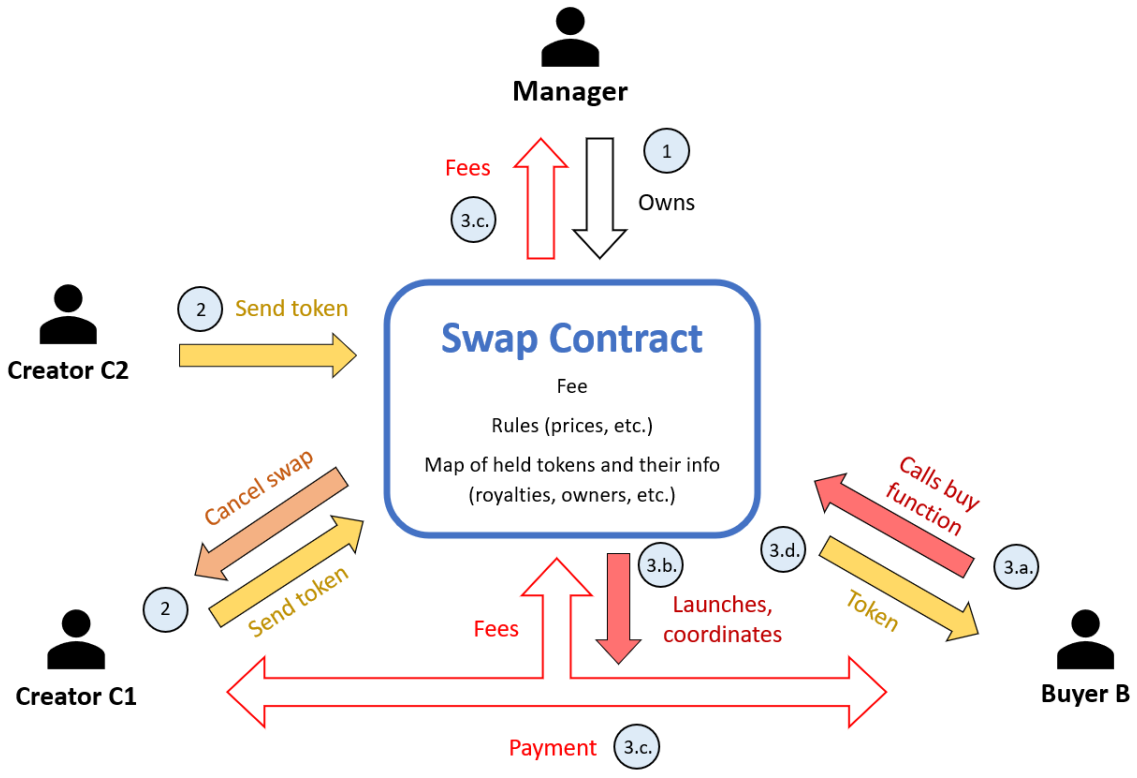

Figure 19: Simplified Swap Contract workflow [GTB22a]. Numbers represent steps, in order; yellow arrows show token movement; red arrows show purchase execution; white and red arrows show royalty distribution.

These operations happen under the hood of user-friendly interfaces. An example of a full implementation can be found in [GTB22a]. The above explanation is simplified and shows the central aspect of the platform. Notably, tokens typically stay on a given marketplace,

allowing for secondary sale traffic to also occur on that marketplace. Secondary sale refers to the sale of a previously owned token that is not directly minted to the buyer. This secondary market is far from negligible. For instance, while the primary market of the *Bored Ape Yacht Club* NFT collection generated 2.2 million USD, it earned its creators 54 million USD in secondary sale revenue via their 2.5% royalty [QAD22]. We shall discuss our advanced solution for NFT royalty distribution in Chapter 4, Section II.

The nuanced definition of tokens in addition to their massive market appeal has led to many ambiguous cases with regards to IPR. For instance, ownership of an underlying asset is sometimes unnecessary in creating or exchanging an NFT. This has caused public outcry on more than one occasion [KSH22] [YTB18] and led to most marketplaces asking sellers to have underlying rights on the represented assets. The fundamental misunderstanding of the nature of tokens even led to a cultural clash jumpstarted by online users "right-clicking" an image represented via NFTs on blockchains as it appeared on image searching engines to save them. The eventual movement of "right clickers" contained people with varying knowledge of blockchains and tokens, and encompassed motives going from provoking ignorant token owners looking for speculative investments to protesting what they believed to be an unfounded market. This ideological debate became a widespread cultural phenomenon from 2019 to 2021, populating online message boards, media headlines, and lawsuits [THO21].

The general market of NFTs, and consequently blockchains, has suffered repeated reputational damages due to malicious actors [KSH22] [DAS22] [ROY23] (cf. Chapter 1, Section III), with controversial news making their way to mainstream, non-specialized media outlets [YTB18]. It is even interesting to note that accomplished careers and online fame have been achieved investigating NFT schemes and frauds [MON22] [SOR22]. Even the most prominent actors have acknowledged that NFT abuse is rampant. Namely, *Opensea* (a historic Ethereum marketplace that, according to [BLO23], 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 [MAN22b]. By their admission, over 80% of the NFTs created using this specific process were as such.

While this market monopolizes token-related discussions, it is essential to remember that tokens bear several other forms than art and can enable more complex processes that could require or benefit from different distribution methods than marketplaces. For instance, IPR management can be significantly facilitated through the exchange of designated tokens which can materialize by the management and traceability of patented and copyrighted IP, as investigated by [BAM22]. Other use cases also require content to be monitored or subjected to specific limitations (*e.g.,* royalties), which prevents their distribution through marketplaces and sometimes even justifies the creation of dedicated means of distribution. Further technological opportunities associated with the tokenization of novel assets are studied in [HEI21]. The prevalence of the NFT digital art use case has overshadowed many of these diverse use cases over the past years.

Critical Limitation: Token use cases are not properly specified and their needs are not catered for by the current day distribution schemes.

3.III. Token Protection

As brought forth in the first words of our introduction, the preponderance of visual assets in the current landscape makes for an immense market present in a huge variety of verticals which must face considerable challenges. The hyper-distributed and decentralized paradigms exacerbated these issues, creating a 10-figure market [FBI23] of assets requiring protection. Taking a step back, the fair and systematic distribution of royalties and the establishment of completely transparent IPR was a major argument in the forthcoming of NFTs since their inception [QAD22]. Now that the global NFT market can be counted in tens of billions of USD [FBI23], [QAD22], [BWC22], although no precise statistics report the benefit split, the promise of a sustainable hub for digital artists seems not to have been upheld during this expansion.

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 [OCD20] issued by the Organization for Economic Cooperation and Development (OECD), which highlighted the importance of these assets, particularly when the assets in question represent real-world assets, as we will discuss further in Chapter 4. While the security of the web3 asset within the blockchain environment is assured by a wide range of native blockchain mechanisms and careful private key management, abuse regarding the relationship with the original asset is not guaranteed and scarcely discussed. The case of data storage modification mentioned hereabove is an illustration of such abuse, but subtler and more prevalent ones also exist. Notably, the quasi-systematic absence of fair compensation of royalty holders for token exchanges moves a lot of ink in the web3 space.

Some marketplaces compensate artists for every transfer of their NFT if it occurs on the platform. Specifically, 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. Most of them allow for a predetermined royalty payment to occur on the primary market, which means that an artist only ever receives money for the first sale of the token. If that token is flipped for 5 or 10 times the initial price, the original artist sees none of the revenue. Payouts are also handled diversely, as some marketplaces pay them out in real-time while others stagger payments to save on gas costs. Further royalty rules are marketplace-specific: for instance, *OpenSea* does allow secondary market royalties that the seller, not the buyer, pays. These royalties are far from negligible, as Ethereum marketplace royalties have already added up to almost 2 billion USD [QAD22]. Of course, platforms have no power to enforce their rules once the token has left their borders, leading them to implement solutions meant to encourage further transfers on their services, limiting interoperability.

Marketplace-agnostic standards are starting to emerge in an attempt to create community-supported trusted precedents. Leading this initiative on Ethereum is EIP2981 [BUR20], which provides the closest framework to a fully usable standard for royalty-enabled tokens. The basic idea behind it is to include royalty information in the metadata of a token, as illustrated in Figure 20. According to this solution, the burden of this extra cost should lie on the consumer, and the value of assets should be contingent on royalties and not be considered separately. Metadata information is added after the creation of the

token, as shown in Step 1 Figure 20. After that, the token can be used freely, including being sent to a marketplace Swap Contract (Step 2 Figure 20). Upon purchase, the marketplace can enforce the payment detailed in the metadata (Step 3 Figure 20).

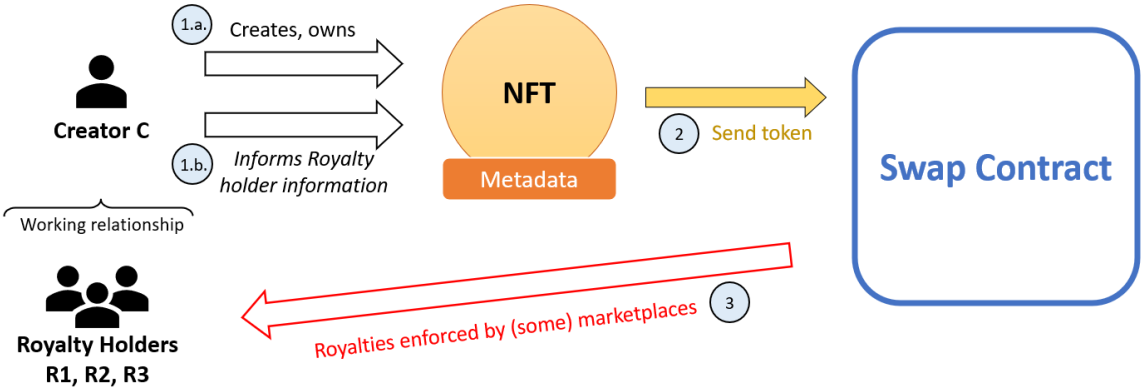

Figure 20: EIP2981 workflow. Numbers represent steps, in order; yellow arrows show token movement; red arrows show purchase execution; white and red arrows show royalty distribution.

Although users advanced unique features and extensions [GTB20], the EIP2981 authors decided to keep nothing but the lowest common denominator not to impact the gas cost of every user, most of which do not need more than the basic functionality. Moreover, complex operations directly affect the standardization process and public acceptance. The simple EIP needed a 12-month validation process, an exceptionally long time on the blockchain calendar. The enforcement of the royalties is not dealt with in the standard and is left to marketplaces. The EIP2981 authors believe artists and the public will eventually pressure marketplaces into adopting the standard, a point from which legal arguments for royalty distribution could be in the realm of possibilities. The *Mintable* [MIN23] and *Coinbase* [COI23] platforms have adopted EIP2981, while *OpenSea* has signaled they will implement it soon. Some upstart exchanges have seized the opportunity to gain market shares by ignoring royalty practices and displaying lower prices. However, the biggest traders with known addresses cannot use them while maintaining their reputation.

Unquestionably, blockchains can pay out royalties systematically and transparently alongside token sales. But a question that sparked passionate debate amongst blockchain actors is “Should they?”. The topic is contentious: While some marketplaces and actors are trying to enable a royalty-friendly environment, others are attempting to dissociate the ideas of royalties and NFTs [QAD22]. Indeed, established examples go both ways: painters typically do not receive compensation past the initial purchase of their work, while music artists usually do. Blockchains tokens lie in a new spot where the answer has neither been defined nor will it be within our study. [THO22] extensively discuss the necessary changes that could enable a healthier environment for IP transfers in blockchains. We would also like to acknowledge that [MAD23] investigated royalty distribution within the context of software licensing NFTs, which can be connected to our subject.

Critical Limitation: Although tokens are protected *per se*, the IPR relating to physical or web2 scarcely transfers to the web3 space.

3.IV. Blockchain interoperability

Before leaning into specifics, it is important to identify why some actors are still reluctant about the blockchain space. The main barriers towards industrial blockchain adoption are identified by [DAV18], a survey cast upon six hundred blockchain executives after Bitcoin's first market surge and during the rise of application-enabled blockchains. It was a pivotal time when companies had to take a stance on blockchain. To "What are the biggest barriers to blockchain adoption?" regulatory concerns unsurprisingly ranked the highest, closely followed by interoperability issues. In the context of this thesis, it is particularly interesting to note that intellectual property concerns ranked as the most popular third reason. The problem, although occasionally clouded by the more endemic issues associated with blockchains, is at the back of many minds. Newer studies [BEL21], [APP22] report interoperability concerns as the most significant limit to blockchain adoption.

These interoperability concerns directly affect the capacity to integrate blockchains in versatile applicative workflows on three major concerns: interoperability within a given blockchain (*e.g.,* between applications supported on one blockchain *i.e.,* ***intra***-blockchain interoperability), interoperability between blockchains (*e.g.,* the capacity to send information and assets between blockchains seamlessly, *i.e.,* ***inter***-blockchain interoperability), and interoperability between blockchains and web2 software (*e.g.,* in the integration of a high abstraction project, *i.e.,* ***extra***-blockchain interoperability). Indeed, when executives mention interoperability, one can presume they are referring to the capacity of blockchain solutions to interoperate with their established workflows. On the other hand, when interoperability is brought up between blockchain actors, it usually refers to the ability of different blockchains to exchange information and assets. Finally, interoperability within a blockchain environment is scarcely addressed as it is often erroneously assumed. We address the current state of these three concepts hereafter.

3.IV.A. *Intra*-blockchain interoperability

Let's take the example of marketplaces, which are surprisingly incompatible given that they are meant to exchange *de facto* standardized assets. Some assets minted through marketplace-specific tools and wizards are designed only to be compatible with their platform of origin. While this is somewhat uncommon, marketplace functionalities are not carried over in most cases. This includes restrictions, royalty payments, metadata, *etc*. The reason for this is recurrent of Section II and has to do with competition. Blockchain asset exchanges are finite, hence all platforms fight for their share. Enabling in-house features to carry over to competitors is simply not in the best interest of the marketplace bookkeepers.

Regarding barriers to adopting royalty-friendly systems and their interoperability, two points can be highlighted for perspective's sake. The first is the frenzy of the NFT market at the time of writing: many blockchain applications are still in a regulatory gray zone, which leads to confrontations between enforcement agencies and some dApps (cf. Chapter 2, Section III). This climate does not provide fertile grounds for building solid and widely accepted regulations. Furthermore, interoperability within a given blockchain is generally a consequence of community-proposed standards, which are naturally slower to emerge than private solutions due to their lifecycle, which must be thought out, discussed, developed, and adjusted before they even need to go through the standardization process, implementation, and acceptance. Initiatives that do not follow this avenue stand little chance of acceptance.

Blockchain Basic: Interoperability within a blockchain is quasi-exclusively guaranteed by applicative standards.

3.IV.B. *Inter*-blockchain interoperability

Interoperability between blockchains is by far the most explored aspect of the three. In this section, we compare various approaches pushed by different entities to provide a snapshot of the literature at the time of writing. We cannot claim exhaustivity as solutions are numerous and constantly emerging. We use [BEL21] as a basis for the census of existing solutions while adding initiatives that have emerged since the publication of the survey. Solutions for connecting blockchains can be classified into three categories: cryptocurrency interoperability, blockchain engines, and blockchain connectors.

a) Cryptocurrency interoperability

The first is centered around the free movement of cryptocurrencies between blockchains. It inherits directly from definitions brought by [BUT14] to include solutions such as sidechains, notary schemes, and Hashed Time-Locks (HTLCs):

- Sidechains are independent blockchains (with their token, protocol, consensus, *etc.*) linked to a mainchain using a two-way bridge. It is akin to what intranets are in the web2 paradigm and typically enables the exchange of assets with the mainchain, sometimes going a step further and bearing some of the mainchain's computational load.
- Notary schemes rely on an entity (typically centralized) to monitor and manage actions between blockchains to provide continuity across frontiers, *e.g.,* when this Smart Contract is deployed on blockchain 1, mint *n* tokens on blockchain 2. Users can typically purchase different cryptocurrencies using their own or fiat

currencies. These exchanges have been targeted by hackers [ABH19], sometimes successfully.

- HTLCs are cryptographic Smart Contracts that serve as time-based escrows. They are an extension of atomic swaps [BLA19], peer-to-peer trading mechanisms often used in DeFi. In HTLCs, parties use their private keys to unlock a hash during a given time frame to unlock the actions of the Smart Contract.

These approaches are also often combined to produce more robust schemes with different compromises in their security and performances.

b) Blockchain engines

Blockchain engines focus on the general ecosystem as opposed to cryptocurrency specifically. The general approach consists of a high-level implementation of a solution that provides flexibility in the implementation of its specifics (*e.g.,* protocol, consensus, Smart Contracts). Information and assets can subsequently be moved between environments supported by the engine.

The EVM is the patient zero for blockchain engines (cf. Chapter 2, II.A.), and it jumpstarted the conversation of blockchain ecosystems breaking the barriers of any given network. It was introduced as a yellow paper (a more technical version of a white paper) in [WOO14]. It is an engine that manages the state of EVM-based blockchains to enable not only the exchange of information and assets but also allows developers to write and deploy Smart Contract across multiple EVM blockchains without significant changes. These Smart Contracts can then interact with other EVM-compatible blockchains through the EVM, allowing anyone to create multi-chain dApps. This aspect is managed by a distributed state machine maintaining transient memory which can execute arbitrary machine code.

[TAK18] provides numerous illustrations of this high abstraction concept, from transaction ordering to memory stacks. Figure 21 shows one of these images illustrating the update of EVM world states (the single state of the EVM at a time *t*).

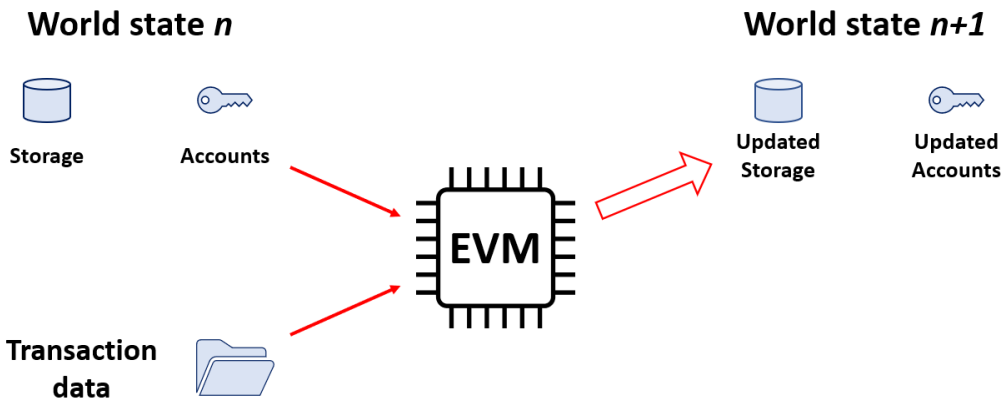

Figure 21: Illustration of how the Ethereum Virtual Machine (EVM) updates blockchain states with new transaction data.

Since the EVM, other engines targeting other visions of interoperability have emerged. While the most publicized feature of Ethereum 2.0 was the shift to a PoS consensus mechanism, it also brought advances in interoperability via a sharding system supported by a new virtual machine (namely, eWASM, introduced in the final phase of the Ethereum upgrade). Sharding consists in dividing the state of the machine (including the history of transactions) into components (shards) that can interact with each other and the Beacon-chain (which supports the coordination of the network) supporting the main chain. Transactions can occur within a shard or between multiple shards, which are compared to "islands that can do their own things" in [LED22]. Ethereum 2.0 is only one way compatible with Ethereum 1.0, with ether from Ethereum 1.0 being compatible with Ethereum 2.0, but not vice versa.

Blockchain engine initiatives have also sprouted outside Ethereum, one of the most important is Polkadot [WOO14], an abstract meta-protocol providing two-way pegs with Ethereum and BTC tokens and bridging with EVM blockchains. The following explanation uses Polkadot-specific terminology, but similar logical elements can be found across other solutions (*e.g.,* Cosmos).

The central entity of the environment is a relay chain that joins all participants and ensures the shared security of the system. Secondary parachains and parathreads, sovereign blockchains with particularities, and use cases, connect to the relay chain to gain interoperation with mainchains. The relay chain itself communicates with main chains using brides. The main advantage of the solution is the capacity for small blockchains with insufficient resources for major interoperability solutions to connect, communicate, and exchange information with mainchains. This also makes scalability one of the strong points of blockchain engines. Blockchain engines' success heavily relies on their governance, as centralized councils and technical committees must ensure the best interest of the overarching network and the cooperation of actors with varying interests.

c) Blockchain connectors

Blockchain connectors operate at a lower conceptual layer and can hence serve blockchains that started operating before the connector appeared. The category is logically much broader and contains a variety of solutions that sometimes bear little to no resemblance to each other. Some examples include trusted relays, blockchain-agnostic protocols, blockchain of blockchains, and blockchain migrators. Briefly:

- Trusted relays redirect transactions from one blockchain to another by acting on behalf of each blockchain on the other's protocol.
- Blockchain-agnostic protocols fundamentally use gateways (akin to payment terminals in the physical world, although the web3 concept goes further in validating cross-chain transactions) to redirect transactions between blockchains appropriately. The paradigm was extended to cross-chain dApp building.
- Blockchain of blockchains are ambiguous frameworks that can be interpreted as connectors and engines. They provide reusable data, network, consensus, *etc*., for creating application-specific blockchains (*e.g.,* parachains) that can connect with mainchains.

- Blockchain Migrators enable end users to migrate the state of a blockchain to another by taking key metrics such as runtimes, write and read times and prices, *etc.*, into account. Specific blockchains must be integrated into the solution to be supported. Smart Contracts cannot be transferred as easily at the time of writing, although they may become in the future.

Blockchain Basic: Inter-blockchain interoperability modules are actively being developed by public and private initiatives but remain centered around specific use cases.

3.IV.C. *Extra*-Blockchain interoperability

Although blockchains have cemented their position by comparison and often opposition to web2, both environments are inextricably linked. Even considering the base function of exchanging cryptocurrencies, users needed to pass through traditional software to access their wallets. Furthermore, application-enabled blockchains enable direct two-way communication with web2 and rely on off-chain storage for asset data (cf. Section II).

Specifically, the rise of applicative blockchains quickly encountered the problem of scalability. The architecture of Nakamoto's blockchain, not having been theorized with high-abstraction software capabilities in mind, could not support the gas cost and network congestion brought by this paradigm shift. Ethereum was the infrastructure that emerged to embrace such applications and offer them to the world. Yet, even after having been created for the purpose of supporting web3 applications, Ethereum has had and continues to fight scalability issues as one of its major challenges. Yet, modifying the underlying protocol to improve its scalability as new opportunities arise is not possible. Hence, higher abstraction level layers are tasked with this assignment. Briefly, *layer 0* refers to common technology powering all blockchain, *layer 1* refers to most common self-sufficient blockchains (*e.g.*, Bitcoin, Ethereum), *layer 2* (the only layer not discussed in Chapter 2) designates networks that are dependent on another "mother" network and typically enable functionalities not supported by *layer 1* [GAN23] [SGU21], and *layer 3* denotes the application layer aiming to connect blockchain with end users.

One of the most important concepts of blockchain *layer 2* are rollups, protocols managing transaction batches off-chain and only committing their final state to Ethereum *layer 1*. This processing is much faster than on-chain validation as it enables the use of off-chain computation, which is many orders of magnitude faster. Their protocol is very complex and has some similarities with the blockchain protocol as they rely on Merkle trees, with a more significant focus on headers. This makes rollups very secure and does not undermine decentralization [THI22]. Two main rollups are currently at the state-of-the-art: optimistic rollups and zero knowledge rollups (zk rollups). While zk rollups submit their transactions with cryptographic proof of validation (ZK-SNARK: Zero-Knowledge Succinct Non-Interactive ARgument of Knowledge or ZK-STARK: Zero-Knowledge Scalable

Transparent Argument of Knowledge) and logically require validation computation (*e.g.,* StarkWare, zkSync), optimistic rollups assume all transactions are valid and hence require very little computation (*e.g.,* Arbitrum, Optimism). A basic overview of rollups is shown in Figure 22. To combat fraudulent transactions, optimistic rollups require collateral[11] from validators and open a window of contestation where anyone can contest the validity of at least one of the batch's transactions. Fraud proof validating the transactions (hence committing the required computation) is then done, and offending validators see their collateral frozen. Validators are incentivized to act in a trusted and efficient manner by earning returns on their collateral if no incidents occur.

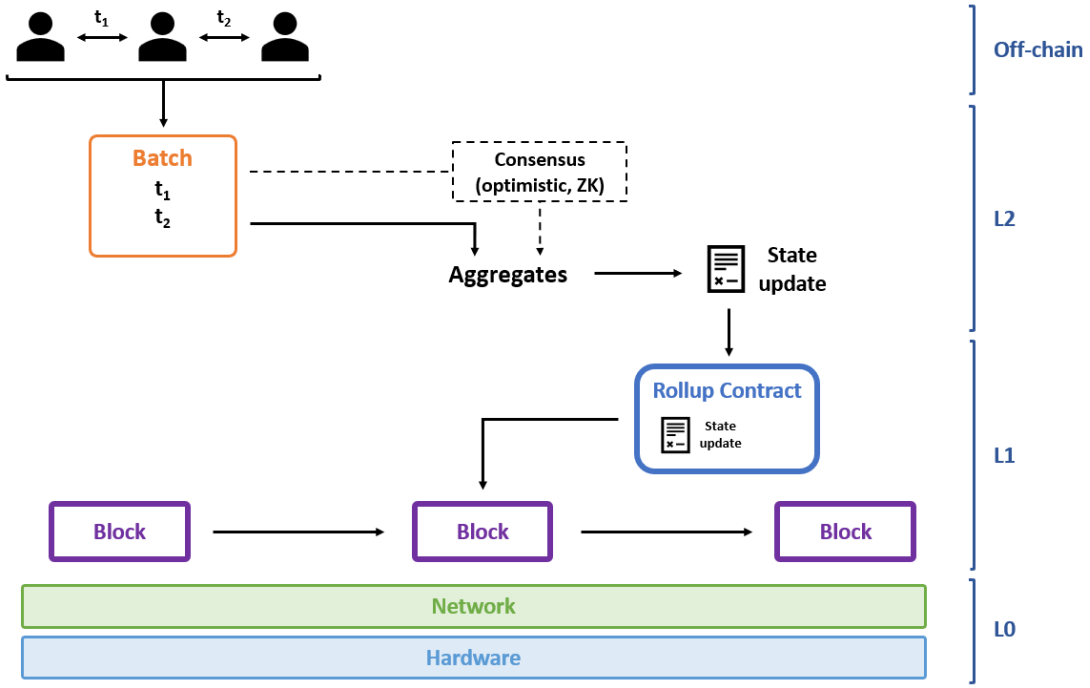

Figure 22: Overview of the general workflow used in optimistic and zk-rollups to increase blockchain transaction processing capabilities.

The above enables the intervention of off-chain computation in low abstraction level operations (namely transactions), but the focus of a large portion of the industry has now shifted to higher abstraction operations (namely Smart Contracts). Smart Contracts can serve as automatic transaction launchers after a pre-programmed condition is met but are now equipped with the capacity to interact with web2 software, which powers a variety of use cases (*e.g.,* DeFi). This connection allows blockchains to access information and ideas that are not possible in a web3-exclusive paradigm, such as real-world (or even live) data or verifiable randomness [LAV22], in addition to making others financially viable (*e.g.*, prediction markets or image processing, cf. Chapter 4, Section III). Private information is often treated off-chain as on-chain data appears publicly.

Web2 solutions have evolved to interact with blockchains and Smart Contracts through specialized libraries. These libraries now enable the launching of cryptocurrency

[11] We intend "collateral" as its financial definition: something pledged as security for repayment of a loan, to be forfeited in the event of a default.

transactions and Smart Contract function execution with specified inputs by connecting to wallets and acting on behalf of end users. As such, one can trigger the execution of a Smart Contract function on demand using Python, Java, or any other language dotted with one of these libraries for their blockchain of choice. While few Smart Contract programming languages can send information back to these libraries, they can emit events that can be caught to trigger further actions. This connection provides the base of dApps as web2 software interfaces very well with human end uses.

Smart Contract capabilities can also benefit from off-chain computation (hence making hybrid Smart Contracts [SOL21]), which they do via oracles, or better, DONs (*e.g.,* Chainlink [BRE21]), which open hybrid computation applications and minimize the trust requirements by compromising on the feature richness of centralized computation. In short, DONs act as an intermediary between web2 and web3 by complementing the information layer of blockchains [CHAI23]. This connection does bring to light the potential issue of a single point of failure where the link itself can be attacked and is often referred to as "the oracle problem" [CAL20]. The general workflow of DONs is shown in Figure 23. More information about DONs can be found in the literature, notably in [EZZ22], which covers the fundamentals of DONs, and in [ALB20], which provides a review and states the open challenges DONs are to face. shows how DONs can enable hybrid Smart Contracts. Further off-chain execution schemes have also emerged in this field [FRA22], notably as an answer to Ethereum's rising fees.

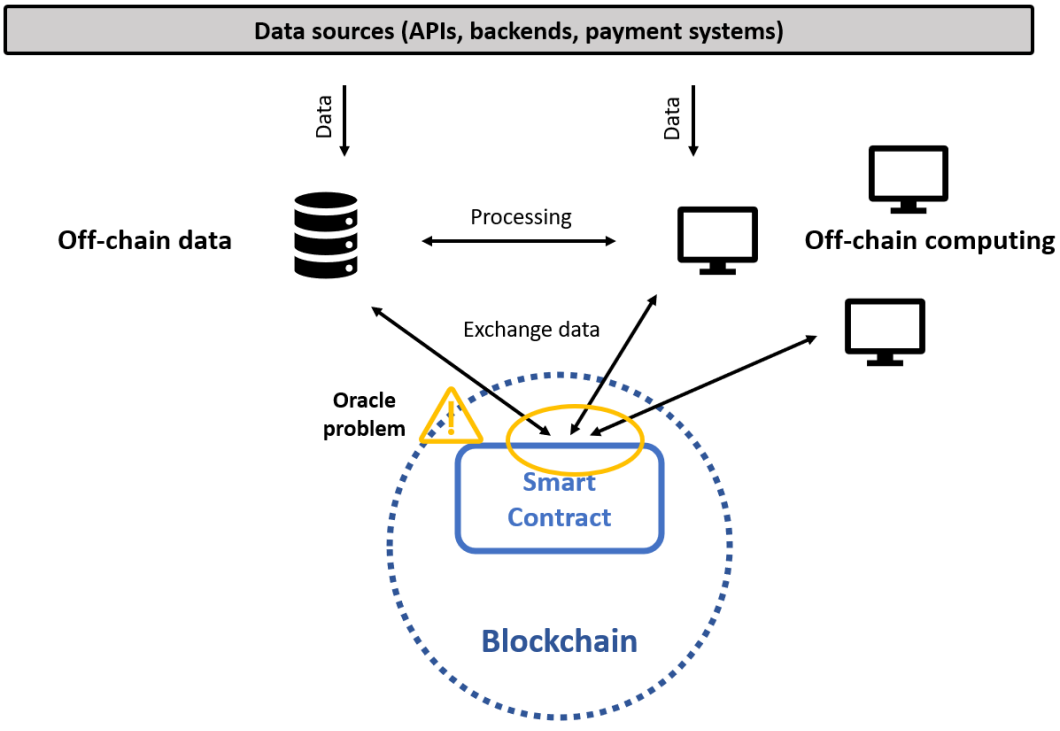

Figure 23: Overview of the general workflow of Decentralized Oracle Networks (DONs), an information layer web2-web3 hybrid solution designed to process data normally unavailable in blockchains. The "oracle problem" single point of failure is highlighted.

Blockchain Basic: Connections between web2 and blockchains enable the broadening and sophistication of use cases but spark the threat of a single point of failure.

3.IV.D. Retrospective view on interoperability

The active literature on *inter*-blockchain interoperability would suggest a multifaceted state-of-the-art that enables anyone to freely move assets and Smart Contracts between blockchains. The reality is much less sparkling. First, the understanding that many solutions mentioned and analyzed by the most thorough survey [BEL21] were acknowledged as endorsed vouches for the competitiveness of a space where each solution and initiative tries to fight for adoption and market shares. Second, the fact that many of the prospected results and mechanisms were published as grey literature[12] suggests that double-blind peer review[13] is not the standard, even for public solutions. Moreover, many of these are destined to solve specific use cases on specified blockchains, undoubtedly driven by underlying economic interests. Consequently, a standard, unified, and trusted environment enabling the interoperation of all blockchain functionalities is still far from the current state.

Yet, we can draw from the current state of the space to formulate ideas about where it is headed. The blockchain connector approach is the most widely used across industries because of its speed of implementation and non-reliance on lengthy consensuses or standardization processes. Considering this fact, we can extrapolate from the use cases that were treated or even considered. Unsurprisingly, DeFi bears cross-chain support using the above approaches and continues to be a driving force in the web3 space. Innovations have taken form in cross-chain payment channels, multi-party swaps, and hybrid decentralized exchanges. Enterprise and business processes also, directly and indirectly, benefit from interoperability solutions and steadily follow the state-of-the-art. Surprisingly, cross-blockchain dApps see the same scale of attention as blockchain migrations, primarily through the hybrid connector approach. This can be attributed to the limit of current blockchain applicative computational capacities. Advances that will overcome these limitations will likely result in a significant increase in cross-chain Smart Contracts, in turn supporting dApps. Blockchain migration has seen the support of the very influential Hyperledger effort through Hyperledger Cactus, promising asset migration for consortiums through blockchain migration.

Although solutions with different fundamental philosophies might be more adapted to some currently neglected use cases, this short analysis provides insight into the priority uses that are raising investments and accelerating the field.

Key Issue: Blockchains interoperate poorly within themselves, between each other, and with web2 applications.

Critical Limitation: Blockchain integration into high abstraction architectures and workflows is made challenging by the multi-level lack of convenient and efficient integration solutions.

[12] "Grey literature" refers to information produced outside of traditional publishing channels. This includes reports, government documents, evaluations, etc. which are often not submitted to third-party peer review.
[13] Third party reviews where nor the authors nor the reviewers know each other's name or affiliations.

3.V. Off-chain tools used in this thesis

The rest of this thesis will advance methodologies that operate in relationship with prior technologies from a variety of web2 domains. While we cannot provide a complete state-of-the-art analysis of each vertical, this section aims to give the necessary base understanding of the well documented "legacy" solutions we will use. We will not discuss the theory backing these technologies but their practical applications. First, we briefly discuss archetypal programming which we will use to uniformly format data. Then, we introduce two ISO/IEC standard notions: Media Ontologies and Internet of Media Things (IoMT) as they will intervene in our solutions. Finally, we introduce the basics of visual content protection and discuss their prior associations with blockchains given that the protection of multimedia assets is one of the critical limitations we address in this thesis.

3.V.A. Archetypal programming

Some technological bricks we will employ will use simple forms of prototype-based programming, which we will further refer to as prototyping (it is referred to as classless or instance-based programming in the literature). Prototyping is a higher abstraction form of standard object-oriented programming which ignores the notion of class to focus on archetypal objects [LAU99]. It enables the creation of objects that directly inherit their properties (e.g., methods, variables) to other objects. Different interpretations of more advanced principles define specific prototyping approaches, but the need for systematic inheritance or low-detail characterization usually drives the objective. One of the defining factors of any given prototyping approach lies in the inheritance model it bears. Typical delegation is often supplemented by the inclusion of new or the exclusion of old properties and myriad other logical links [DON92]. Figure 24 shows a simplified view where a prototype generally characterizes a set of houses.

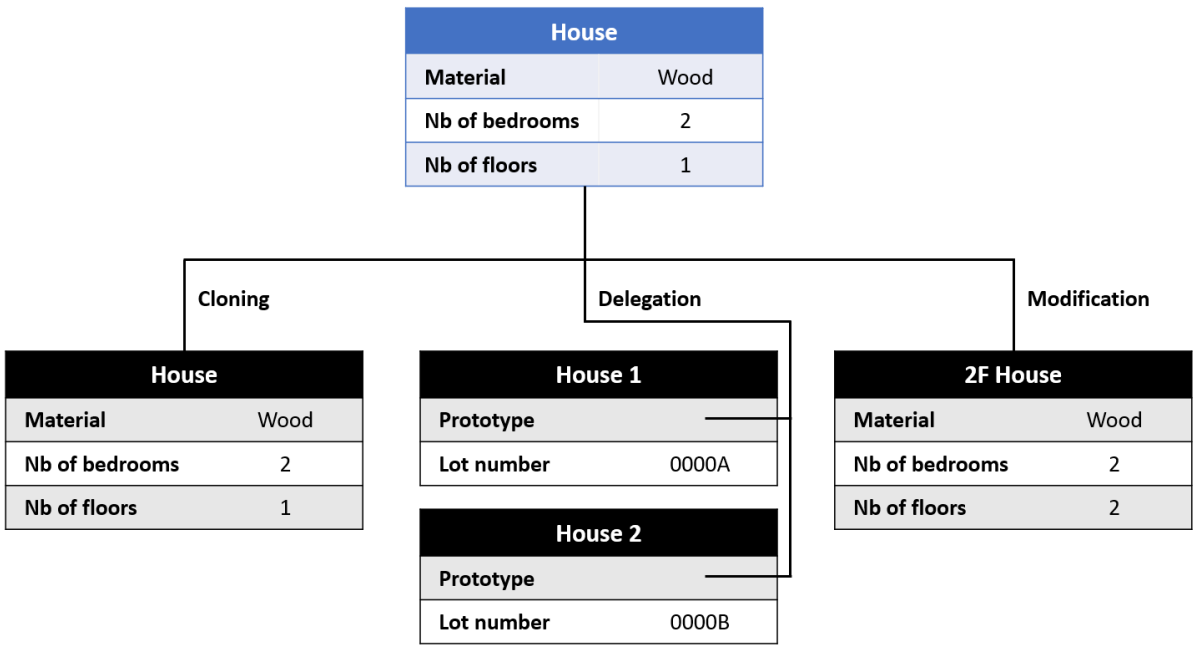

Figure 24: A basic overview of archetypal abstraction, featuring cloning (which copies data), delegation (which generalizes sets of properties), and modification (which changes certain properties).

In a software sense, prototyping usually lets one look at overarching problems directly with little deviation caused by hard-to-interpret code. It also enables generalizing specific examples into abstract principles, which can jumpstart processes with tangible patterns. While the most widely used prototype language is JavaScript [MDN23], it is preceded and accompanied by numerous other languages, including Moostrap [BOO23], Agora [AGO23], Newton-script [NTS23], *etc*. [NOB99] [DON92] provide a general analysis of the concept and its applications while [DON98] attempts the empirical classification of prototype-based languages.

In Chapter 4, we will use archetypal programming that lies somewhere in the gray zone of prototyping on two occasions: to uniformly identify assets across heterogeneous environments and systematically create code that has been formalized elsewhere.

3.V.B. ISO/IEC 21000 Ontologies

In the context of ISO/IEC 21000, MPEG developed a set of standardized schemas for the codification of intellectual property (IP) rights for media in CEL and MVO [ISO16a] [ISO17], two machine-readable ontologies detailed hereafter. The Contract Expression Language (CEL) [ROD09] [JAN10] is an XML language for representing media contracts, while the Media Contract Ontology (MCO) [ROD16] is a language for describing media contracts as ontologies with RDF (OWL). IPR ontologies also include the Media Value Chain Ontology (MVCO) [ROD09], which facilitates rights tracking for the fair, timely, and transparent payment of royalties, and the Audio Value Chain (AVCO) [LEG15], which extends MVCO for audio applications. These contracts enable prior affirmation of IP dispositions and connect four logical entities:

- Smart Contract for Media: the resulting Smart Contract that manages the original contractual rights on the blockchain.
- Parties: Contractual parties that are eventually represented by blockchain accounts and transacted through addresses.
- IPEntity: IP-driven assets represented by NFTs.
- Obligations: Contractual obligations, permissions, and prohibitions. NFTs can also represent them.

These ontologies were developed to support the execution of rights-related workflows in environments such as blockchains that can serve as IP registries and execute contractual rights systematically and transparently. By making these formats directly translatable into Smart Contracts, clauses can be executed seamlessly, and contractual data can be transferred from one blockchain environment to another, contributing to *inter*-blockchain interoperability. Figure 25 shows extracts of a CEL and MCO Contract, giving examples of a specified party, statement, and obligation.

```
<cel-core:Party id="licensor">
    <cel-core:Person>
        <cel-core:Name>AG</cel-core:Name>
        <cel-core:Details>Aggregator AG</cel-core:Details>
    </cel-core:Person>
</cel-core:Party>

<cel-core:Statement>
    <cel-core:Subject partyRef="revenue1"/>
    <cel-core:Act>
        <rel-r:possessProperty/>
    </cel-core:Act>
    <cel-core:Object>
        <cel-core:Item name="MC">
            <dii:Identifier>isan:1234mca</dii:Identifier>
        </cel-core:Item>
    </cel-core:Object>
</cel-core:Statement>
```

```
DeonticStructuredClause:
Clause ID:  p926
Deontic type:  Permission
Subject:  RAI
Object:  {'name': 'Title1', 'Identifier': 'isan:ab234yz'}
Act:  communicationToThePublic
Issuer:  licensor
Runs:  {'number': '6', 'hasValidity': 'P0Y0M7DT0H0M'}
Means:  ['satellite']
Is Exclusive:  true
Fact Composition:  factIntersection
Language:  it
Location:  ['VA', 'IT', 'SM']
Temporal interval:  2012-11-18T23:59:59
2009-11-19T00:00:00
Access Policy:  freeOfCharge
Delivery Modality:  broadcasting
```

Figure 25: Contract Expression Language (CEL) extracts showing parties, statements, and obligations, respectively (top), and extract of a Media Contractual Object (MCO) generated from a complete CEL input (bottom).

The specifications of these formats can jumpstart the execution of underlying rights in controlled environments such as blockchains, which birthed ISO/IEC 21000-23 Smart Contracts for Media. This emerging standard provides APIs for converting these XML and RDF media contracts to Smart Contracts (on any blockchain, as the process is largely agnostic with respect to the environment) and is the working group we participated in.

3.V.C. ISO/IEC 23093 IoMT

Internet of Things, a term coined by Kevin Ashton in 1999 (based on Mark Weiser, Bill Joy), describes a machine-to-machine (M2M) communication paradigm where various physical devices interact with the physical world through sensors and connect through the Internet. It superseded and reshaped the Internet of People era that saw the explosion of social networks to connect the heterogenous world that surrounds us by using a mix of technologies such as Wireless sensor networks, RFID, NFCs, *etc*. The concept is now firmly implanted in numerous aspects of our daily lives through smart appliances and smartphones, but also within the industry where Internet of Things (IoT) largely contributes to process automation, condition controls (*e.g.,* temperature, pressure), *etc*. Today, Cisco accounts for approximately 3.6 networked devices per capita [CIS20], and half of network connections being M2M. Figure 26 shows the data brought forth in their annual reports between 2018 and 2023.

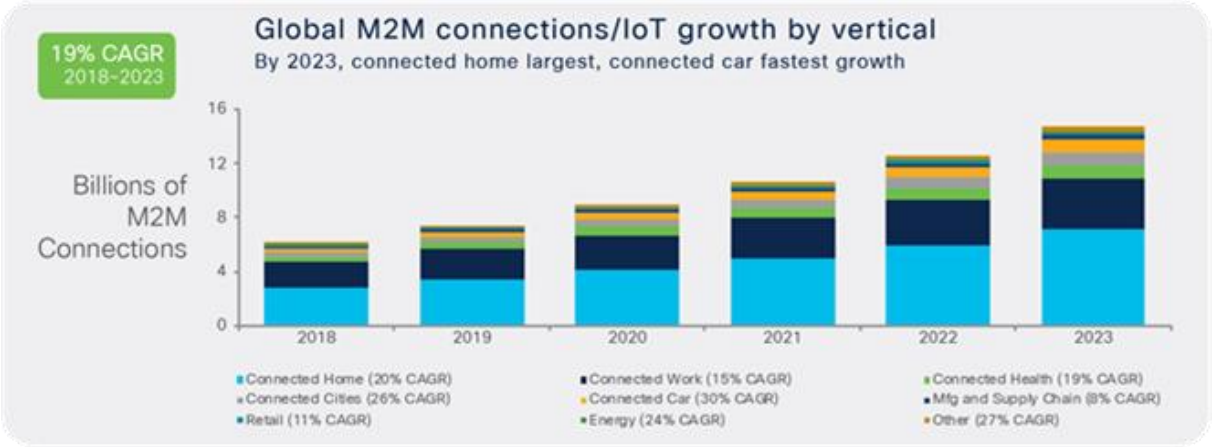

Figure 26: Global machine-to-machine (M2M) connections between 2018 and 2023 as reported by Cisco [CIS20].

The domain is now standardized by ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission), pillars for specialized standardization across various industries. These organizations comprise national bodies, technical committees, and working groups that tackle diverse fields and activities. These bodies also collaborate with governmental and non-governmental organizations and with one another. For instance, the ISO/IEC-related work further in this thesis was related to the joint technical committee dedicated to information technology, ISO/IEC JTC 1, and more specifically within the Moving Pictures Expert Group (MPEG, *aka* ISO/IEC JTC1 SC29).

Specifically, within the field of IoT, media-centric applications and services can offer the provision, interpretation, representation, or even analysis of multimedia content collected by media devices such as cameras, microphones, vibro-haptic devices, *etc*. MPEG introduced the notion of Media Thing (MThing), which is defined as a Thing capable of sensing, acquiring, actuating, or processing media content or metadata related to such content. Some examples of MThings include MCameras, MMicrophones, MDispalys, MAnalysers, or MStorages. IoMT is currently under the scope of the ISO/IEC 23093

Internet of Media Things standard family [MPG19][ISO21a][ISO21b][ISO22b] and is now known as MPEG-IoMT.

MPEG-IoMT ensures interoperability among mediacentric applications and services designed and deployed for interpreting, representing, or analyzing multimedia content collected by MThings. It provides an architecture, specifies APIs, and details compressed representations of data flowing between MThings [ISO21a][ISO21b][ISO22b]. Various MThings can thus be designed, orchestrated, and operated in multiple tasks such as acquisition, rendering, processing, or multimedia content storage. This facilitates the design and implementation of complex systems capable of processing massive multimedia content, thus bridging the gap between user expectancies and their feasibility. Our interest in the subject was accompanied by our active participation in ISO/IEC JTC1 SC29 WG7 and the publication of various standard inputs, as presented in our extended abstract.

3.V.D. Visual content protection

Visual content has been prevalent since the emergence of web2 and continues to occupy more and more space in the daily lives of billions of people and a wide range of industries (*e.g.,* social networks [CAO1], uncrewed vehicles [CHE19a]). 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 [GVR23b]. While multimedia assets represent a foundation of the modern digital economy, they can still suffer from improper use or IPR infringements, such as copying, illicit commercial exploitation, resource waste, or false appropriation [ABB22]. The online piracy market was estimated at 51.6 billion USD in 2022 [KOS20], and abuses have an immediate, tangible influence on fields such as the movie industry [MA14].

Regardless, no matter the specific application and throughout its entire lifecycle, visual content protection is currently ensured by a large variety of conventional solutions, ranging from data encryption (which ensures the privacy of data during transmission and storage [NAD05]), digital signatures (which track the content by compact digests of its digital representation [KAT10]), watermarking (which tracks the content by inserting additional data inside it [POD01]), to digital fingerprinting (tracking the very semantics of the content by compact digests of its human perceived features [LU09]).

While some of the methodologies we shall bring forth in Chapter 4 could benefit from data encryption at the level of data storage or hidden signatures without any extra drawbacks, it focuses on content tracking. Indeed, in the process of turning multimedia content into web3 assets, said content naturally becomes accessible and hence vulnerable to copying or abusive usage. As such, the first step to enabling content tracking is by precisely identifying it.

Cryptographic hashing functions [SOB12] might seem like a natural solution to identify content through fixed-length digests, allowing one to look for copies throughout data

storages efficiently. Yet, 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 piece. 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 their digital representation and, as such, would have different cryptographic hashes, although we would commonly call all three the "same picture." The same logic applies to operations such as recording, broadcasting, *etc*. This idea births the notion of near-duplicated content, which [LIU13] provides four definitions for. The most useful to our applications 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).*

Consequently, cryptographic hashing functions sensitive to slight modifications and producing uncorrelated digests from correlated inputs cannot be used for our purposes. In other words, cryptographic hashing is not robust to near-duplication. This is useful when multimedia inputs are static and not meant to be manipulated further as any unintended, however slight, modification to the input would completely modify cryptographic hashes.

In opposition to cryptographic hashing, near-duplicated content tracking (also referred to as *similarity-preserving visual fingerprinting*, *visual fingerprinting*, *robust hashing*, or *perceptual hashing*) is a methodological framework that considers the semantic content of its input when creating a digest. Visual fingerprinting functions feature four stages: the *transformation* stage, which applies spatial and frequency transformations to the input to prepare its features for the second stage; the *feature extraction* stage, which isolates features and sometimes selects the most relevant; the *quantization* stage, which forms an intermediary hash of bytes; and the *compression and encryption* phase, which outputs a short perceptual hash [HAD12]. These perceptual hashes hence contain precise information about their original inputs while allowing for a level of distortion in subsequent verifications. Consequently, each hash has a given set of images that it can relate back to. This means these functions can be used to identify slightly modified versions of multimedia content. As such, the bit-by-bit equality used in cryptographic hashing leaves its place to similarity measures such as like normalized correlations or the Hamming distance to check how similar perceptual hashes and their original inputs are. This is illustrated in Figure 27. This similarity-preserving feature allows fingerprints to be used to find nearest semantic neighbors of an input amongst a given dataset.

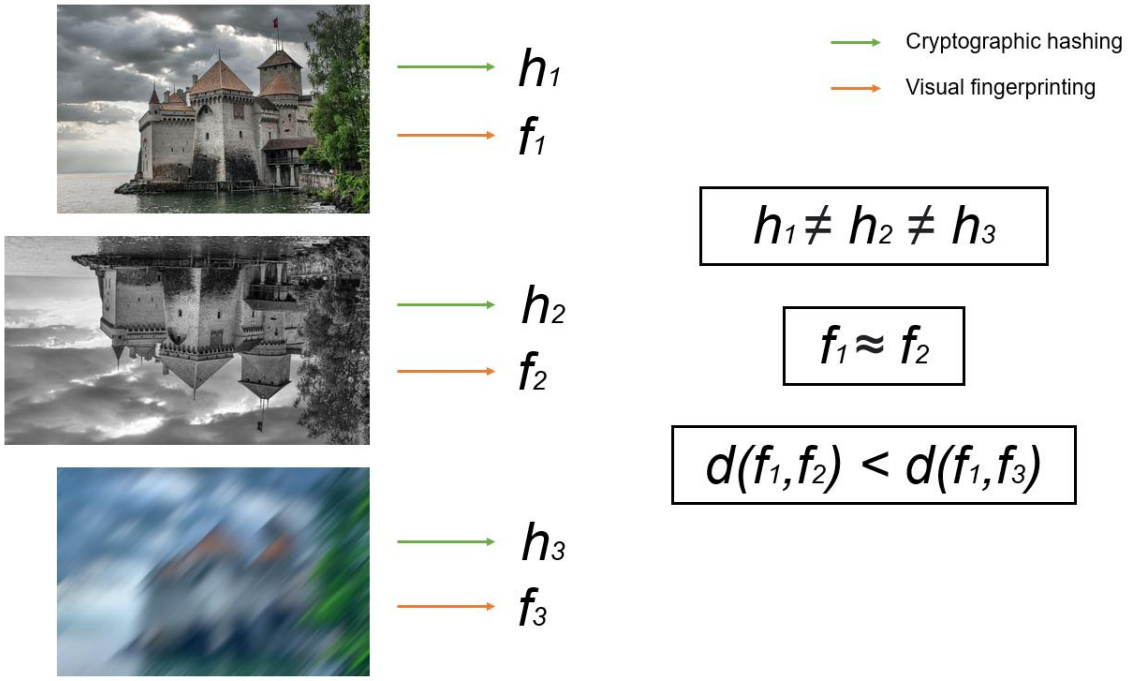

Figure 27: Illustration of the differences between digests produced by cryptographic hashes and similarity preserving visual fingerprints and illustration of the notion of distance between fingerprints.

The broader characteristics of visual fingerprinting methods diverge significantly. For instance, the input types of fingerprinting methods can be images, image sequences, videos, audio, *etc.*, and the format of the digest identifying the semantic content can range from short strings to large matrices. These differences can be explained by the targeted use case and performances, *i.e.,* the amount of semantic information required to identify an input with the desired resistance to modifications or robustness. Once a method and similarity measure (*e.g.,* normalized correlation, Hamming distance) is decided upon, a similarity threshold is selected to reflect the chosen near-copy detection sensitivity. A low sensitivity will consider mildly different inputs unique, while a high sensitivity will more precisely flag minor differences. These differences may be semantic or a result of content modifications or attacks. These attacks can be malicious or the result of multimedia operations or regular use. Each method bears different levels of robustness to various attacks. While some fingerprinting methods might not perform well after the resizing of an input, others might suffer from luminosity or color balance changes.

Given these characteristics, visual fingerprints can be used in advanced feature-based comparisons [OOS02], using a variety of multimedia formats as inputs (*e.g.,* audio files [SAR09]). Yet, their applications are not limited to feature detection and can range from Digital Rights Management [CHE19] to network protection [CON04]. Their primary use remains video identification, for which various methods have emerged. [ALL22] presents a survey of the landscape of fingerprinting for video files while [JIA16] 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 [WU04]. Visual fingerprinting has also repeatedly been associated with deep and machine learning to be trained and optimized over specific

sets of data [HE16][JIN21][KOR17][KRI12]. Some fingerprinting methods allow for various input formats, while some include metadata and instance data in their methodology.

While many research efforts have focused on the quest of feature extraction, few address the security of the hashing system (some examples include [FRI00] or [VEN00], which integrate private-key cryptography directly within the process). In this thesis, blockchain will be doing the heavy lifting to ensure levels of security and integrity expected when dealing with multimedia content. It is important to understand that perceptual and cryptographic hashing are tools that do not solve the same problem. In Chapter 4, we will use both to identify near-copies of visual content and shorten invariant digests for easy storage, respectively. For readability's sake, we will abbreviate "visual fingerprinting" to "fingerprinting" in the rest of the thesis.

3.V.E. Prior associations between blockchain and media applications

Multimedia content has had to face novel issues on web3 environments [QUR20] (*e.g.*, metaverse content creation [CHE22] [DON22]). These issues present wide ranges of challenges including environmental [XU22] economic [BUH23], and social [DUA21]. Hence, many public and private initiatives have emerged, often taking the stance of adapting proven web2 solutions to the web3 paradigm. While this approach is valid for some widespread protection methods, it is not compatible with every method. For instance, although fingerprinting methods could add multimedia processing to the blockchain while benefiting from the environment's security, they are prohibitively complex to be computed on-chain. We will address this issue from different perspectives in Chapter 4, hence discussing prior associations hereafter.

When it comes to multimedia content being used in blockchain contexts, NFTs are the central concept in representing such assets (cf. Chapter 2 Section II, Chapter 3 Section II and Section III). Further, multimedia processing can be enhanced via blockchain applicative technologies as part of the process [LI21] or hand in hand with off-chain technology [FRA20]. The joint uses of content protection techniques and blockchains are summarized in [QUR20]. This holistic survey cites encryption, watermarking, and transaction tracking fingerprinting, indicating that near-copy detection using visual fingerprinting techniques had not been associated with blockchain before [MOR23a].

The idea of data storage being used alongside blockchains is not novel. For instance, databases and blockchains were used in an IoT use case in [TSE20] and a cloud computing study in [LIA17]. To the best of our knowledge, the strategy of a replicated hashed "shadow" on-chain database as an integrity verifier, which we use in Chapter 4, Sections III, brought forward by ourselves in [MOR22], was novel. The concept was cemented as a

journal paper in [MOR23a] and extended to provide the basis of our full lifecycle asset management model in [MOR23b], which we present in Chapter 4, Section IV.

Chapter 4. Methodology

This chapter will bring forth this thesis' methodological contributions. It will begin by formalizing the sets of requirements our work shall follow in light of our state-of-the-art analysis. It will then explain our four base methodologies from a macro perspective. Then, we will provide the synergistic combination of these solutions, highlighting what each can bring to complex high-abstraction solutions.

4.I. Key limitations and design requirements

Before moving into our methodological contributions which address the critical limitations we identified in the environment, we establish their foundations in light of the gaps in the current state-of-the-art, as identified in Chapter 2 and Chapter 3. To do so, we identify two sets of design requirements from our previous analysis that we will apply to our Smart Contract-centric and token-centric solutions. These requirements will ensure the methodologies provide added value in their functionalities, do not suffer from technical shortcomings, and enable multifaceted interoperability.

4.I.A. Smart Contract design requirements

The biggest issue facing Smart Contracts is their natural assimilation with web2 software. Further, this association is limited because both target different objectives and are subject to different limitations, as detailed in Chapter 2 and Chapter 3. In short, Smart Contracts functionalities must be simple and summerly implemented to avoid becoming very expensive via their gas consumption or becoming vulnerable. We shall abide by this common driving thought in applicative blockchains and set as the **1st Requirement for our Smart Contract solutions that they shall be simple and lightweight**.

Smart Contract functionalities can be furthered via a wide range of solutions explored throughout Chapter 3, Section I (*e.g.,* DONs). Yet, many of these solutions introduce new threats and points of failure which are yet to be proven defeated. Furthermore, some of these solutions do require the inclusion of third parties, which sometimes reside and operate off-chain. Although these solutions have their place for specific use cases (*e.g.,* the need to retrieve real-time data on-chain), we will focus on self-sufficient infrastructure that relies on blockchain as the sole, zero-trust third party. Our solutions will nonetheless still be compatible with these approaches. Consequently, the **2nd Requirement for our Smart Contract solutions shall impose the absence of further third parties in the off-chain/on-chain connection.**

Taking a step of perspective, one must remember that the absence of a regulated and interoperable blockchain environment constitutes the most significant barrier to blockchain adoption (cf. Chapter 3, Section IV). Unfortunately, regulatory efforts introduce severe latency in the development process, as potential solutions need to be designed, discussed publicly, developed, and adjusted before they even start their standardization and acceptance phases. This latency sometimes opens regulatory gaps. Yet, these gaps are tackled actively by the specialized groups and consortiums blockchain environments have long relied on for regulatory efforts and *de facto* standardization. The absence of standards targeted at specific issues can often be attributed to the fact that the community is in the process of establishing best practices. Intermediary solutions consequently do not have much space unless they can interoperate with emerging standards, designed according to current-day best practices and implemented standards, *i.e.,* backward compatibility has a strong correlation to forward compatibility. Given this, we set as the **3rd Requirement for our Smart Contract solutions that they shall feature backward compatiblity with existing standards.**

Finally, although various solutions enabling the passing of assets from one blockchain to another have appeared, we identified the lack of systematic bridges for Smart Contracts between different blockchains as a roadblock to web3 interoperability. Yet, to not isolate our efforts and potentially close them off to further development in the blockchain ecosystem, we shall make sure that our methodologies can be implemented on any application-enabled blockchain supporting a Turing-complete programming language. The **4th Requirement for our Smart Contract solutions is that they shall be powered by functionalities found across applicative blockchains.**

Design Requirements: Our Smart Contract solutions shall answer to three design requirements imposed by the current state-of-the-art. These requirements, further referred to as **SCDRs (Smart Contract Design Requirements)**, are:

- **SCDR1:** Our Smart Contracts shall remain simple and lightweight.
- **SCDR2:** Our Smart Contracts shall not require extra third parties to operate.
- **SCDR3:** Our Smart Contracts shall be backward compatible with existing standards.
- **SCDR4:** Our Smart Contracts shall be implementable on any applicative blockchain.

4.I.B. Token design requirements

Our token-based solutions will implement Smart Contract bricks within their architecture and inherit the design requirements we specified above. Further, they will abide by extra requirements driven by the current state-of-the-art.

We identified token ownership as a surprisingly cloudy and misunderstood aspect of daily blockchain operations and a fundamental problem when complex use cases related to web2 or physical environments involve tokens and their underlying rights (cf. Chapter 3, Section 2). Connecting web3 assets to legacy media assets is often done to ensure the continuity of operations between these environments. Hence, the **1st Requirement we set for our token-related solutions is that they shall allow users to express their respective IPR and royalties rules before the first transaction.**

Furthermore, asset exchanges are subject to arbitrary limitations due to marketplaces and their lack of interoperability (cf. Chapter 3, Section IV). This price is paid because of the marketplace ecosystem's competitiveness and convenience, which enables creators and owners to send their tokens and specify their price through convenient dApps without worrying about hands-on operations. These advantages carry their penchant for the need to trust the marketplace (or have the technical ability to check their trust mechanisms), the compensation of this third party via a fee, and the seller is bound by all the rules and limitations the marketplace desires to impose. Furthermore, marketplaces cannot enforce creator limitations past the initial transaction of the token. Considering the technical and applicative heterogeneity of current-day marketplaces, the **2nd Requirement we set is that our token-based solutions shall be agnostic with regards to marketplaces.**

There are two approaches to furthering token capabilities and features. The first consists of implementing new software as part of standards in the very source code of token contracts. The second consists of furthering native features using Smart Contracts to manipulate tokens. Having a separate entity handling functionalities could sound like a lackluster solution compared to the immutability of having these functionalities implemented within the token itself. Yet, it allows for flexibility for buyers and sellers that is impossible with unmodifiable hardcoded constraints. Indeed, token modifications must be standardized and adopted, which imposes a lengthy time cycle and a threshold of mass appeal, given that new functionalities increase gas costs for all users. Furthermore, a significant limitation in advanced token features is the impossibility of differentiating sales from transfers. Users can and do move assets, which turns the systematic enforcement of rules that would apply to the sale of an asset into a limited solution. Many have proposed checks to solve the issue, but unfortunately, all provided exceptions that would immediately be exploited by malicious actors looking to bypass rule enforcement. Hence, only a deep protocol engrained in the logic of the blockchain itself might be able to bypass this issue. This potential approach would add a level of centralization that would undermine the very existence of some blockchains. And even then, what would prevent actors from coordinating off-chain *via* an escrow service and misidentifying a sale for a transfer? This situation makes standard rule enforcement very complicated to approach from an *intra*-token perspective. Smart Contract solutions, contrarily, can be diverse and emerge fast, bringing innovation at a rhythm token-centric solutions cannot support. Yet, Smart contract-based solutions only see the passage of tokens, which means any feature they provide they provide is finite, which makes rules easy to skirt. The use cases we shall tackle require high flexibility and the ability to indeterminately impose rules that should apply to targeted tokens. This sets our **3rd Requirement that our approach shall ensure the flexibility to implement custom features that can apply to tokens indeterminately.**

To provide flexible token-centric features, a challenging compromise to make, some have sacrificed the enforceable nature of their rules. An example is EIP2981, which we covered in Chapter 3. The approach is good, as it deals with common token roadblocks but targets blockchain actors in marketplaces. An opposite approach, leaning into blockchain's systematic enforcement targeted at non-blockchain actors wanting to use the environment's leading qualities while keeping the high flexibility imposed by our 3rd Requirement, has yet to emerge. Going back to the EIP2981 example, users suggested many flexible features that could make the solution more adapted to particular use cases [GTB20], yet the authors decided to keep nothing but the lowest common denominator in order not to impact the gas cost of every user, most of which did not need more than the most basic functionality. This was even more important for a targeted standard, as complex features heavily affect the standardization process in terms of length and acceptance. Some upstart exchanges seized the opportunity of EIP2981 adoption to get some market shares by ignoring royalty practices and displaying lower prices, which was only somewhat balanced by potential reputational damages to large traders using them. This does not worry the EIP2981 authors or the community, but we want our solution to avoid this issue. These facts invite us to target enforceability constraints to enable our advancements to be usable without supplementary trust left in third parties. Targeting the

enforcement and flexibility gap identified in the state-of-the-art, we set as our **4th Requirement for our token-based solutions that they shall provide the systematic enforcement of their underlying rules rather than leaving this enforcement to a third party.**

Tokens represent a significant portion of the *de facto* standards the community produces and relies on. The most widely used standard on Ethereum, bar none, is the ERC20 Fungible Token standard. Hence, we can and should extend SCDR3 to apply to the solutions we will bring forth for tokens. Consequently, the **5th Requirement for our token-based solutions is that they shall be backward compatible with already existing standards.**

Design Requirements: Our solutions dealing with tokens shall answer to five key design requirements, further referred to as **TDRs (Token Design Requirements)**, namely:

- **TDR1:** Our solutions should allow all parties to ascertain their positions before any transactions.
- **TDR2:** Our solutions shall be self-sustained, i.e., should not require the intervention of other solutions.
- **TDR3:** When applicable, our solutions shall ensure they affect relevant parties and assets indeterminately or until further agreements.
- **TDR4:** Our solutions shall enforce underlying rules systematically.
- **TDR5:** Our solutions shall be backward compatible with existing standards and state-of-the-art approaches.

4.II. Advanced token-centric solutions

4.II.A. Token broker

Whilst our solution could be used within a variety of contexts, we explain its conception and illustrate it in the context of IoMT, as it supplies a proven foundation and multi-faceted nuance to explore. Within the IoMT framework, devices that do not require excessive security are put online where they can be susceptible to payment counterfeiting, malicious control, access, interception, and redirection [WIL17]. Fortunately, technical solutions exist, but require complex combinations of security solutions and willingness of end users to investigate the security of their devices [SAN15]. For devices which aim to distribute content, the blockchain framework is a very coherent solution which can simultaneously handle access control, provide data immutability, and zero-trust transactions.

IoMT standards make provisions for the use of blockchain solutions in conjunction with MThings, which constitutes our starting point. For instance, the standard already specifies the APIs and the usage of digital coins or legal tender to ensure the payment of MThings for their use, as detailed further for the case of an MCamera (similar mechanisms exist for every MThing). Yet, these considerations were made without concerns about blockchain development, scalable interoperability, and resource consumption. More importantly, it assumes the proper delivery of content without zero-trust mechanisms, leaving much of the power on the side of MThings to the detriment of users. As such, we design a methodology to manage MThing client interfacing more robustly and systematically and implement it.

Before delving into the specifics of the solution, it is critical to understand how MThing APIs are interfaced. Figure 28 shows the sequence diagram between a user and an MCamera as specified by [ISO21b].

To watch a video captured from a camera, the user inquires about the cost per minute for that MCamera, using the standard API with the desired media token defined by `tokenType` (*e.g.,* cryptocurrency or legal tender) and `tokenName` (*e.g.,* Bitcoin or US dollar). The MCamera returns the cost per minute to let the user access the live video using the `getVideoURL()` function. If the user agrees to the price, they ask for a wallet address of the desired type of currency (*i.e.,* MToken). Again, the MCamera responds with the proper wallet address to which the user sends the agreed amount of MTokens through the blockchain, which returns a transaction ID (`tid`)used to confirm the good standing of the payment. With the transaction confirmed, the user can ask for the video stream service to the MCamera using `getVideoURL(tid)`. Before granting access, the MCamera uses check-`TransactionCompletion(tid)` to check the completion of the transaction and the number of received MTokens. Then, the MCamera returns a video URL and streams the video according to user rights. Details about getVideoURL, `CostPerMinute()`, and `getVideoURL ()` can be found in ISO/IEC 23093-3, while the details of `getWalletAddress()`, `sendTokens()`, and `checkTransactionCompletion()` are described in ISO/IEC 23093-2. This interaction enables the Smart Contract to be powered by a simple if/and answering logic per **SCDR1**.

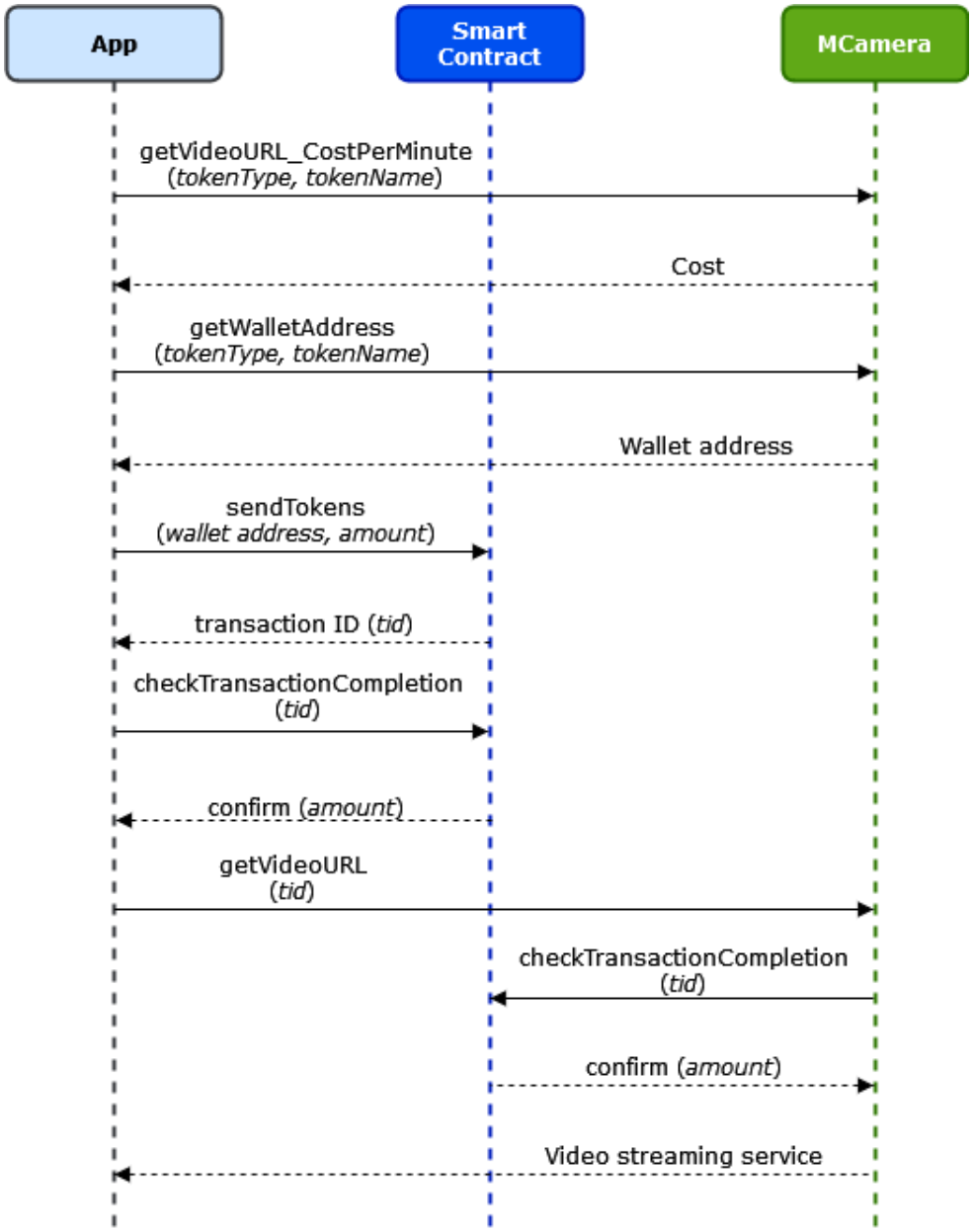

Figure 28: Sequence diagram of the interaction between a user and a blockchain enabled MCamera as defined by ISO/IEC 23093-2 [ISO21b].

We shall keep this information flow and remain under the umbrella provided by MPEG-IoMT while trying to overcome some of its limitations. The biggest has to do with scale: IoT devices rarely operate independently and are conceived to share data formatted to be processed systematically. However, the blockchain APIs do not reflect this philosophy, only enabling the static interaction with a single MThing with its output. The flip side of this shortcoming is that the workflow leaves no margin for error or misuse while not requiring external intervention. We shall keep this advantage while uniformizing the process across MThings, making multi-device management easier and less resource consuming. First, we specify the three parties in context.

MThings:

The MThings retain the exact blockchain representation as in MPEG-IoMT. They are represented by Smart Contracts deployed by designated wallets that define and manage the flow of the MThings' data, as per **TDR1** (*e.g.,* the supply and price of access tokens). Instead of communicating directly with the user, it connects with a Broker.

Broker:

The Broker acts as a vending machine and escrow, communicating with potential buyers (Users) through the APIs provided by MPEG-IoMT (Figure 28), independently from marketplaces, as per **TDR2**. When an MThing is instantiated in the Broker, tokens are minted to represent the services provided by said MThing through a single mixed token standard Token Contract, which is called directly by the Broker. These tokens are then sold to users, and subsequent funds are held according to the MThing's provisions until proof of service delivery. This mode of functioning makes it adhere to **TDR3** and **TDR4**. Finally, **TDR5** is covered using standardized tokens and APIs, ensuring interoperability with contingent use cases. Further, as shown in Figure 33 and Figure 34, the Broker directly interacts with both parties, which makes it abide by **SCDR2**.

Users:

Users can purchase tokens directly from the Broker through the sequence shown in Figure 28, specifying which MThing they want to access data from. Once they have paid for their service, they will be issued tokens that can be redeemed for the specific MThing for the service.

MPEG-IoMT does not make provisions for the modalities of content distribution. As such, we decided not to impose extra constraints on the standards and leave the specifics to MThing operators. Alongside the tokens, the Broker will provide a URL (as specified by MPEG-IoMT) where the user can redeem his new tokens. Once the tokens are redeemed, the Broker is sent proof of the transaction (*e.g.,* the tid or the original tokens). It then frees the funds, sends them to the MThing's Smart Contract, and burns the involved tokens. This escrow mechanism allows content delivery to be made in a zero-trust fashion, as it should be in blockchain environments. In short, the process occurs in four phases: the setup, the purchase, the service delivery, and the verification. This diagram showcases that the Broker's actions are standard Smart Contract operations per **SCDR3**. These actions are limited to data storage, transactions, and token operations, which are features of all applicative blockchains, making this architecture agree with **SCDR4**.

With the parties established, we can now focus on the successive actions that occur between said parties. The idea behind this solution is simple and design-driven: making a common interface to multiple MThings. This interface handles the interaction with users, payments, and asset exchanges. The workflow englobes interactions between three main components: the MThings, the centralized Broker, and the Users. The asynchronous workflow is described in Figure 33 and can be decomposed in four phases detailed hereafter.

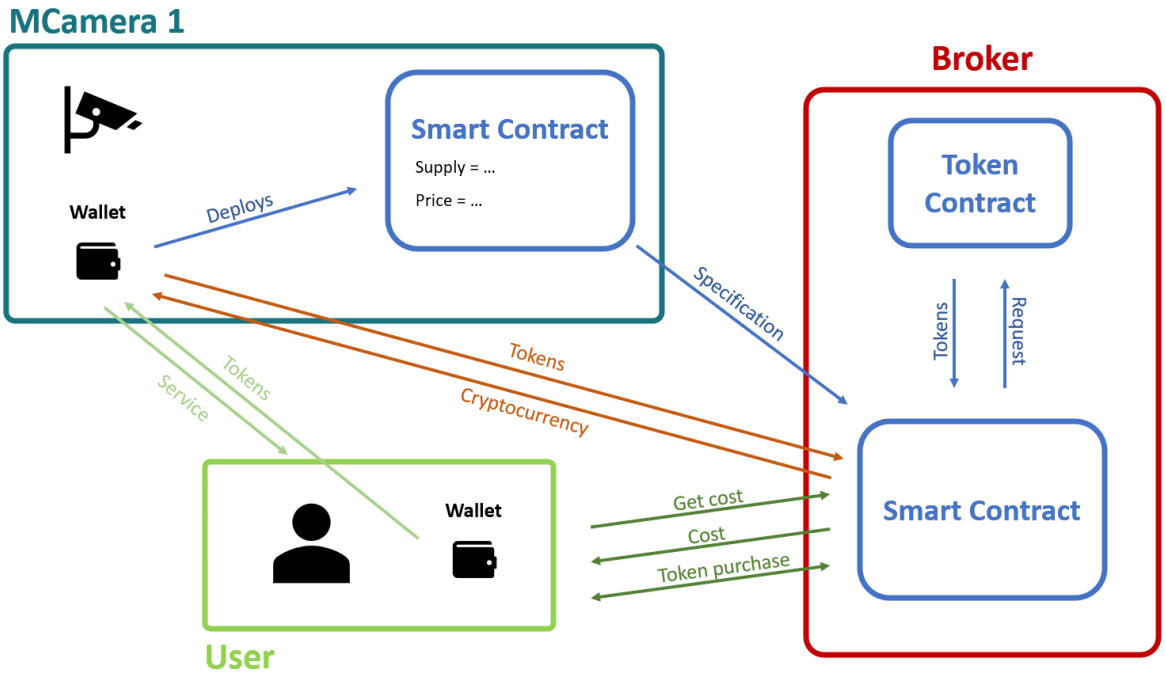

Figure 29: General architecture of the Internet of Media Things (IoMT) broker, showing the setup between MThing and Broker (in blue), purchase between User and Broker (in dark green), content delivery between MThing and User (in light green), and final compensation between the MThing and Broker (in orange).

In the first phase, the wallet associated with an MThing deploys a Smart Contract that will manage the flow of information. This Smart Contract then informs the broker of its specification, which leads to the creation of an appropriate amount of fungible access tokens. This is shown in Figure 31.

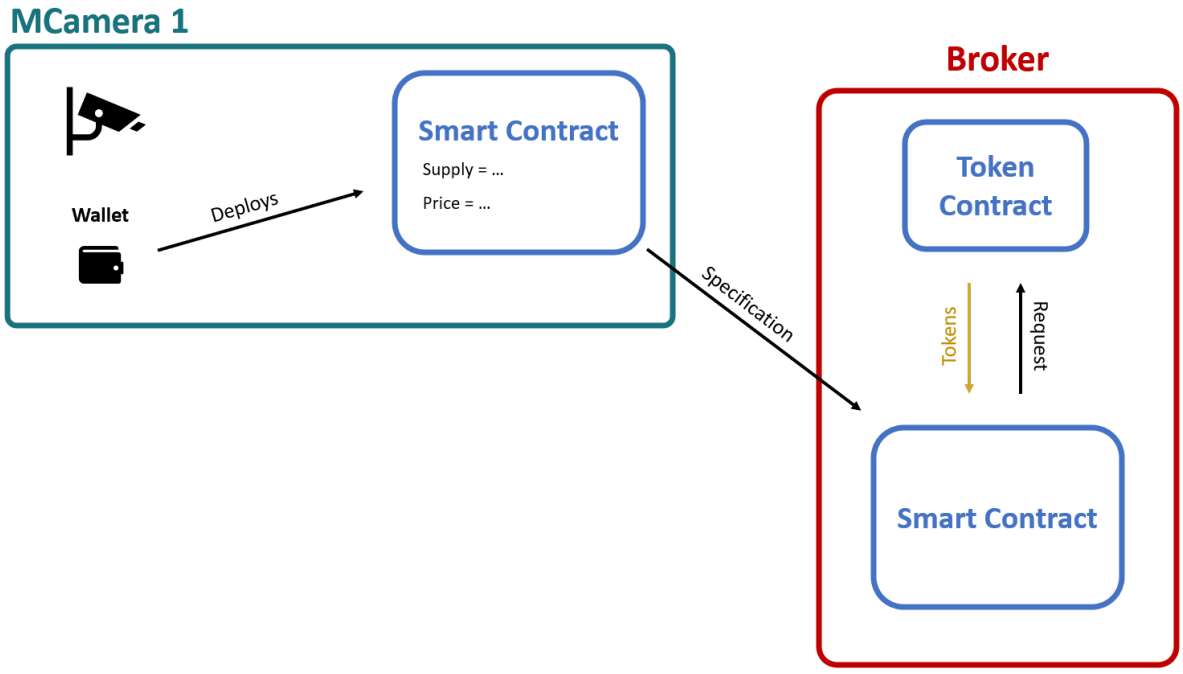

Figure 30: The setup phase of the workflow consists in the MThing setting up its Smart Contract and coordinating with the Broker.

The second phase no longer involves the MThing but a user purchasing access tokens from the broker directly through a dedicated API (cf. Figure 28) as illustrated in Figure 31.

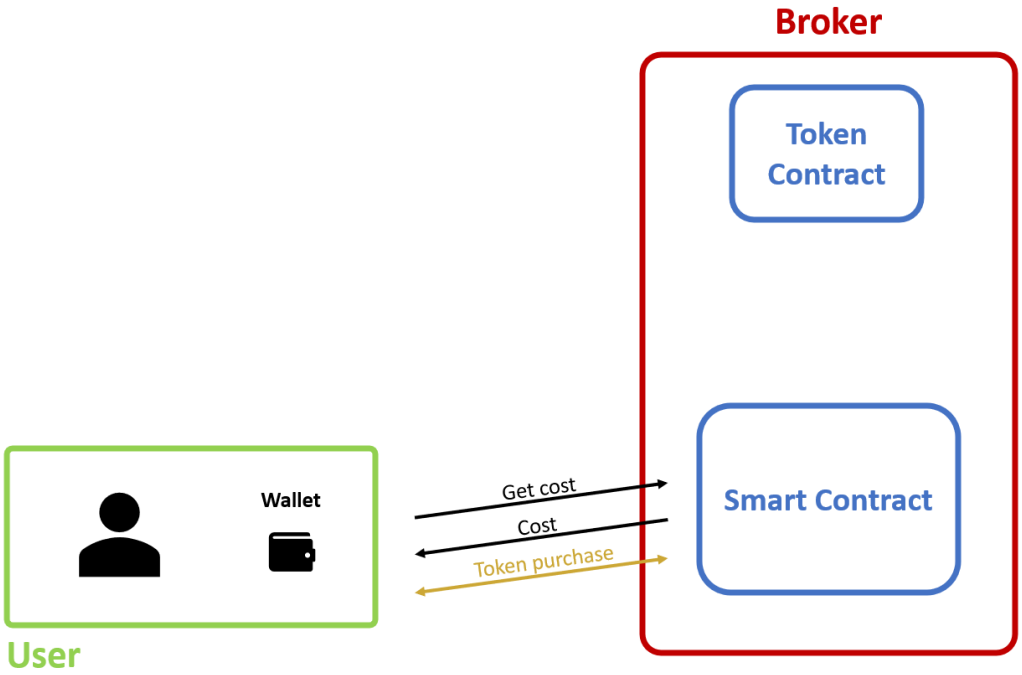

Figure 31: The purchase phase of the workflow consists in a User purchasing tokens from the Broker through an API

Whenever a user wishes to access the data or service provided by the MThing in context, they may redeem their tokens to the service provider directly through a dedicated service. This interaction is shown in Figure 32.

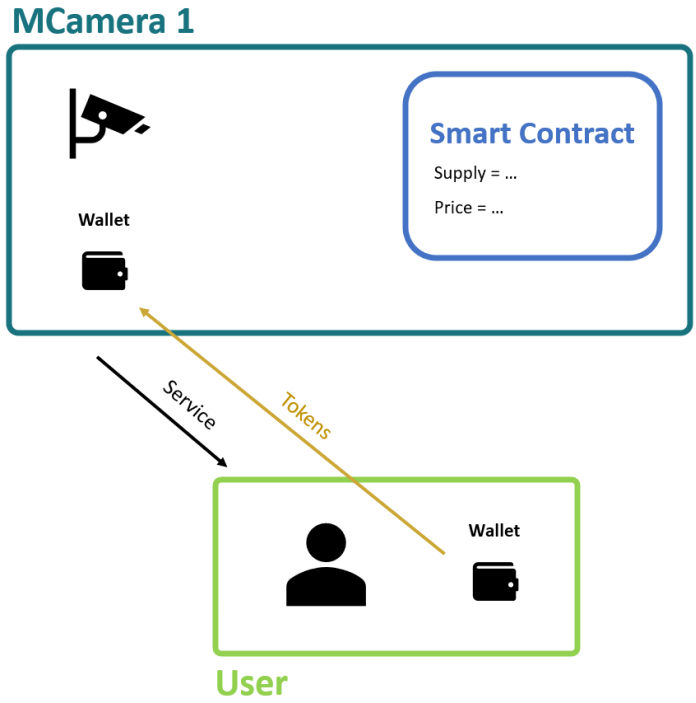

Figure 32: In the third phase of this workflow, a User gains access to the data or service provided by the MThing using their access tokens.

Finally, the MThing may request its compensation by sending the tokens spent by various users to the Broker. This may be done in a staggered fashion to minimize transaction costs, at the discretion of the MThing operator. One such operation is illustrated in Figure 33.

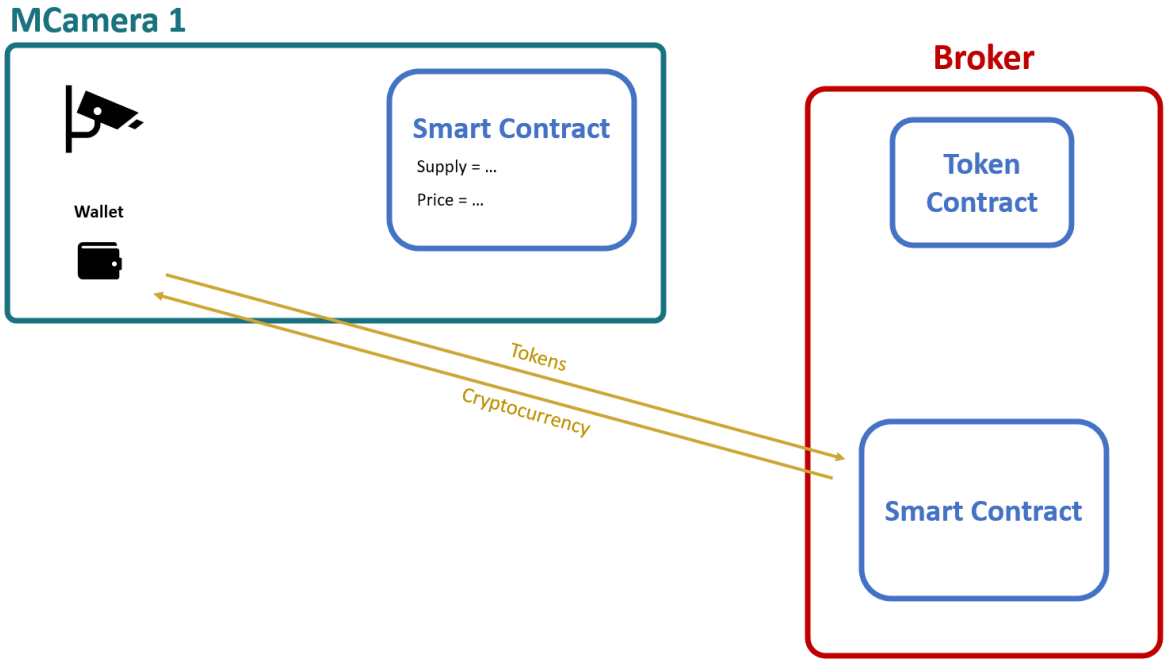

Figure 33: The fourth and final phase of the workflow sees the escrow payment of the Broker in exchange for the access tokens collected by the MThing.

Figure 34 shows the process sequence diagram from a macro perspective, considering the Token Contract as a subpart of the Broker.

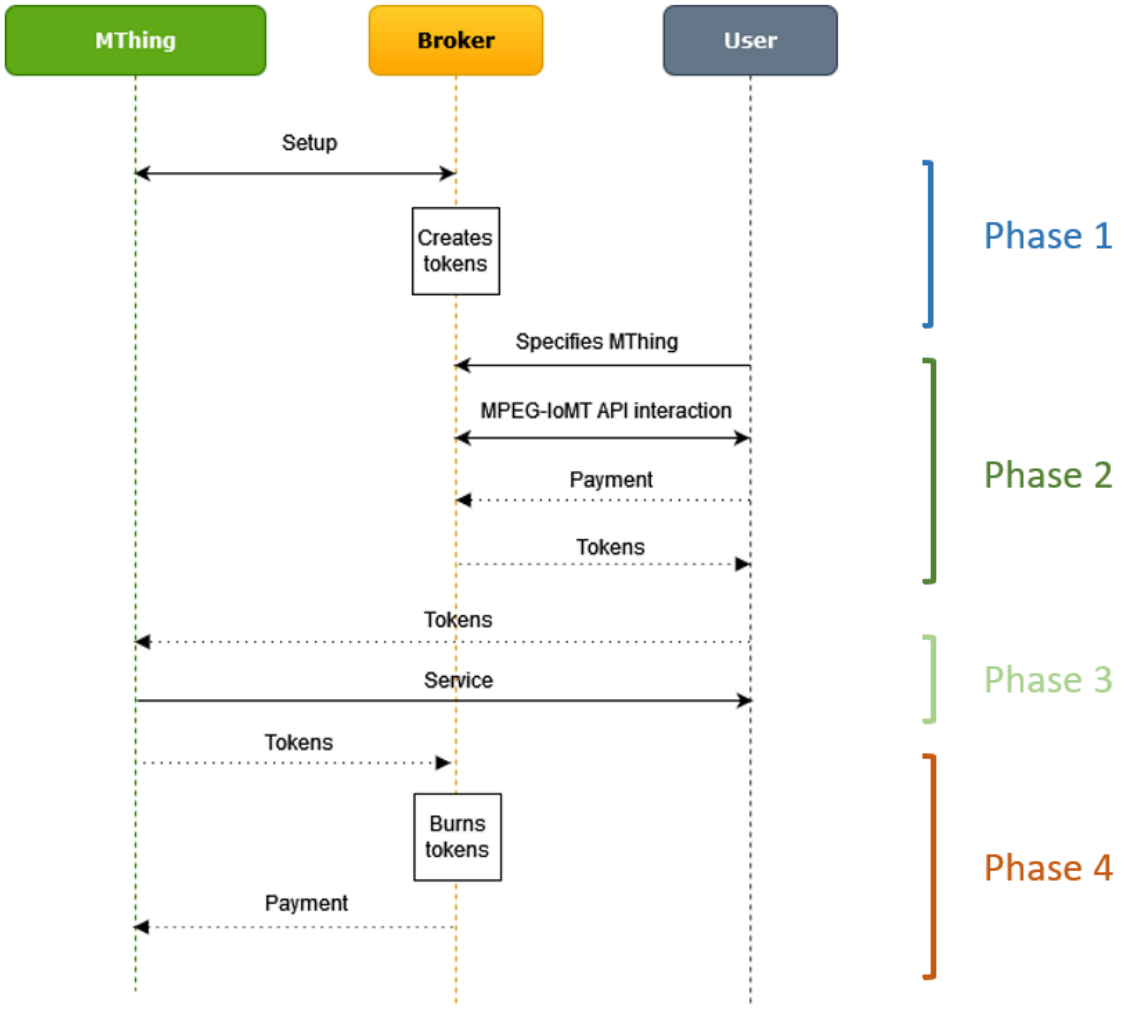

Figure 34: Sequence diagram of the Internet of Media Things (IoMT) broker, showing the asynchronous setup between MThing and Broker, purchase between User and Broker, and content delivery between MThing and User.

This architecture was designed to handle a multitude of MThings simultaneously. It does so by simply minting unique tokens identified by the addresses of the respective Mthing Smart Contracts through its Token Contract. The mixed standard of this contract enables it to support different distribution schemes, from FTs representing a specific duration of access to NFTs giving determined rights. Making a single Broker the point of entry for all MThings not only enables saving resources (in the deployment of new Smart and Token Contracts) but also enables a more effortless process for users that can be reassured the entity they are dealing with directly manages every aspect of the process in a transparent and verifiable fashion.

Table II provides a point-by-point comparison of this solution against its most relevant state-of-the-art counterparts: the MPEG-IoMT provisions and standard token marketplaces. The first two deal with MPEG-IoMT dispositions, the following three with standard blockchain criteria, and the last two with the unique advantages of our solution, namely the escrow enforcement and multifaceted MThing management we provide. Green checks signify compliance and red crosses noncompliance. An implementation of this

methodology can be found in Chapter 5, and a thorough analysis of its advantages, limitations, and further efforts can be found in Chapter 6.

TABLE II
COMPARATIVE SUMMARY OF THE ADVANCED BROKERAGE SOLUTION AGAINST RELEVANT STATE-OF-THE-ART EQUIVALENTS

	State-of-the-art solutions		**Advanced solution**
	[ISO22a]	[GTB22a]	
Standard MPEG APIs	✓	✗	✓
Flexible rule implementation	✓	✗	✓
Charges fees	✗	✓	✗
Zero trust purchase	✓	✓	✓
Zero trust delivery	✗	✓	✓
Escrow enforcement	✗	✗	✓
Can manage multiple MThings	✗	✗	✓

Methodological Solution: The Token broker presented hereabove is an all-in-one solution for the blockchain integration of access control use cases.

4.II.B. Token Level Smart Contract

A retrospective look at Chapter 3 shows that an interoperable, fair, transparent, and scalable way to exchange assets would alleviate most concerns new and to-be blockchain adopters have. Furthermore, the absence of a uniform and reliable *modus operandi* to guarantee the rigorous application of IPRs and the systematic and fair compensation of appropriate parties in web3 prevents the establishment of the trusted global environment aimed for by blockchains. Hence, specifying, designing, and programming an automatic, transparent, and trustworthy enforcement and compensation ecosystem is one of the significant challenges blockchain actors shall answer during the coming years. This challenge will require significant technical resources and answers to philosophical differences between actors, which need time to take form.

Although a long-term goal would be to provide an *inter*-blockchain interoperable ecosystem, short-term efforts shall be targeted at improving interoperability at the level

of individual blockchains. Although initiatives tackling IPR and royalty distribution questions have slowly emerged on large blockchains (*e.g.*, EIP2981, cf. Chapter 3), they have yet to feature indeterminate and enforced solutions that leave control to artists rather than marketplaces, which sometimes have vested interests not to act in an ethical manner. In the following pages, we propose an intermediate step by advancing a fully interoperable *intra*-blockchain solution, allowing for the fair exchange of assets within an ecosystem.

As explained in Section I. A., adding features such as royalty distribution or IPR rules to tokens is possible in two ways: within a standard (token-based) or at the ownership level (Smart Contract-based). The first would rely on a series of actions launched via the `transfer()` function of the token, while the latter would launch those actions externally, *i.e.,* via a Smart Contract. Fundamentally, we must be able to apply restrictions to token purchases and enable sub-transactions to occur systematically while balancing reliability, transparency, and efficiency. We also need to consider past and future, ensuring backward compatibility and the flexibility to ensure interoperability with future developments. On the one hand, token standard-based solutions are limited in flexibility, and solving their main pitfall (the sell/transfer loophole) undermines decentralization. On the other hand, Smart Contracts, although the *de facto* solution, can only follow part of the life cycle of tokens. To circumvent both issues, we specified and designed a synergetic approach, further referred to as the *TLSC – Token-Level Smart Contract*.

The TLSC requires a paradigm shift in token exchanges: as opposed to managing the tokens at the level of the seller (*e.g.,* through marketplaces), the TLSC is initialized with a set of rules and is exchanged alongside the token to follow it during its entire life cycle, enforcing the rules intended by the creator, as illustrated in Figure 35. This way, **TDRs 1**, **2**, and **3** are met. Note that the TLSC also includes retraction rights management to provide a proper lifecycle to the Smart Contract, as dictated by **TDR3**.

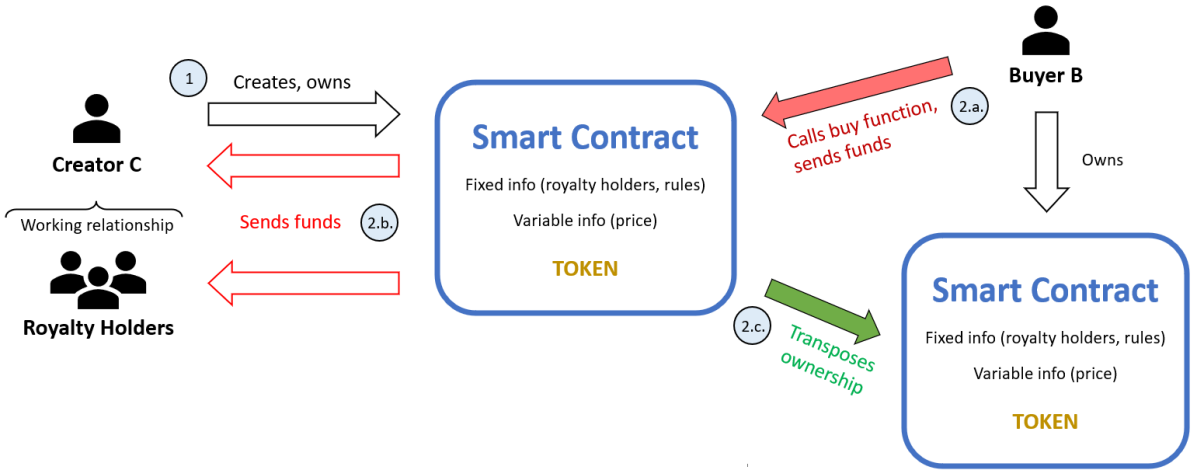

Figure 35: Overview of the advanced Token Level Smart Contract (TLSC). Numbers represent steps, in order; white arrows show ownership; red arrows show purchase execution; white and red arrows show royalty distribution; green arrows show changing ofhands of the TLSC.

Specifically, we decided to use TLSC as a capsule, owning the token and ensuring every transaction follows its initialized rules to satisfy **TDR4**. In alignment with **TDR5** and **SCDR3**, the heart of TLSC uses the functionalities brought by current-day working standards. To ensure enforceability (**TDR4**), we must prevent the bypassing of rules by malicious users simply calling the standard *transfer* method. We elected to change the owner instead because modifying this cornerstone function would render the token standard non-compliant (which would contradict TDR5 and SCDR3). In essence, the TLSC acts as a ‘‘mini swap contract’’ belonging to the current token holder, acting as a zero-trust third party in the purchase of the underlying asset. This third party can apply any limitation or condition initially set by the creator, making this solution completely flexible (**TDR1**). The lifecycle of TLSC encompasses three steps:

- *Initialization*: This phase aims to create the TLSC with the desired set of rules.
- *Trading*: In this phase, the TLSC is traded usually.
- *Termination*: This phase allows for a backdoor if the TLSC is no longer an appropriate solution.

The *Initialization phase* consists of instantiating the TLSC with the set of rules to be applied (*e.g.*, ‘‘5% of each transaction of this token goes to this wallet address’’, or “the token’s price shall never exceed 5ETH”) before sending the token to it. To minimize potential mistakes, the TLSC will only accept the specific token it was built for after the token owner explicitly approves its address. This approval is necessary as the TLSC is not allowed to request the token beforehand, as is defined by token standards [VOG15], [ENT18] (**TDR5**). Before the first sale, we remain in the initialization phase, and all the rules are still modifiable by the owner. This sequence of events is illustrated in Figure 36. As Figure 35 and Figure 36 demonstrate, the TLSC is also simple in its operation, as per **SCDR1**, self-sufficient, as per **SCDR2**, and uses standard operations to be found across blockchains, as per **SCDR3** and **SCDR4**.

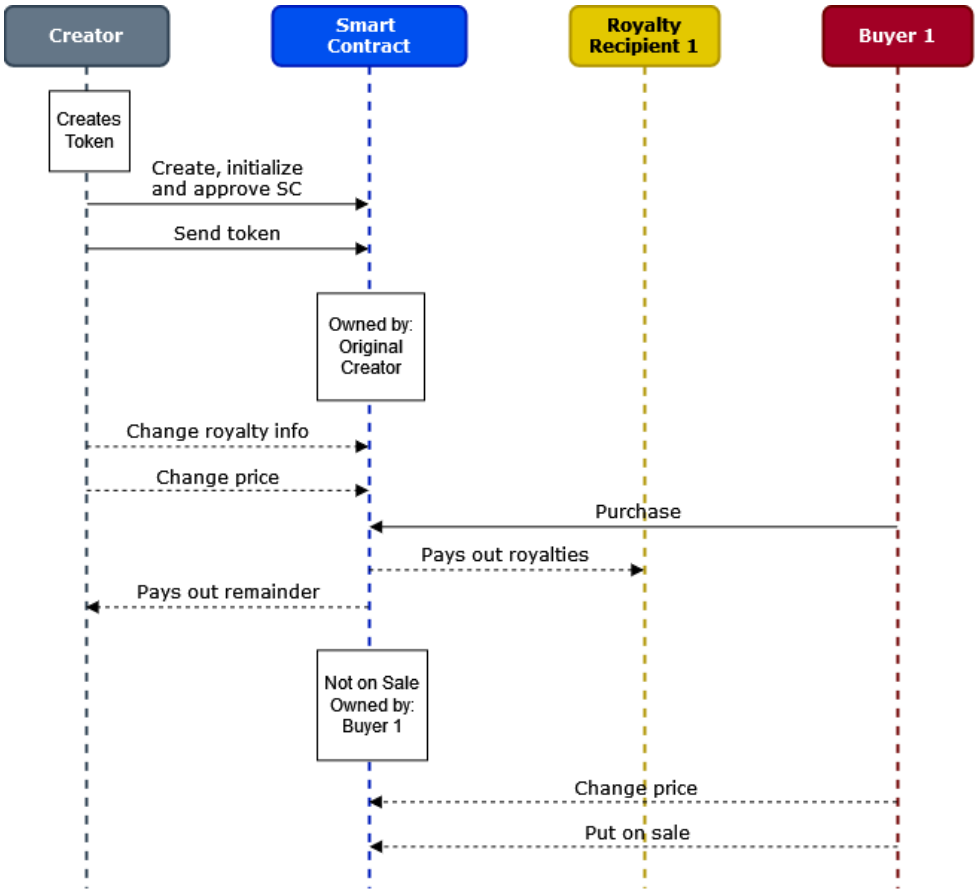

Figure 36: Sequence diagram of the initialization phase of a Royalty Managing Token Level Smart Contract (RM-TLSC), from instantiation to first purchase.

The *Trading phase* begins after the first transfer. From then onwards, the price is the only modifiable field. This means that a token can be flipped for many times, and the prices can vary, but limitations are locked (*Requirement 3*). This philosophy resembles EIP2981 [BUR20] where the value of assets is contingent on rules set out by the creator (*Requirement 1*). The token is considered on sale if and only if the current owner sets a special `onSale` boolean to True. If a token is on sale, a buyer can transact the currently listed price to the Smart Contract with the buy function (*Requirement2*). When one does so, the amount is automatically split amongst the parties listed in the original ruleset and sent, and the owner and sale status are updated (*Requirement 4*). The buyer becomes the new owner and can change the price if they want to. The Smart Contract acts as the NFT, changing hands or resting in someone's possession. Figure 37 shows the *n*th purchase of the token.

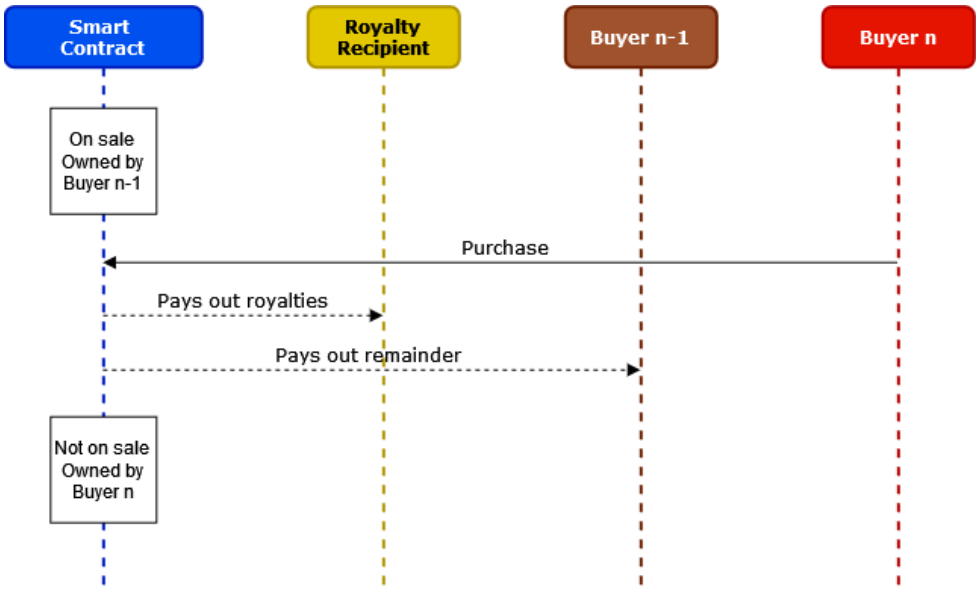

Figure 37: Sequence diagram of the trading phase of the Royalty Managing Token Level Smart Contract (RM-TLSC), during which a royalty payment is executed automatically.

Finally, if an owner wants to extract the token from the TLSC, the retraction rules set by the creator of the Smart Contract (*Requirement 3*) must be followed. In the example below, the current owner can call the *termination* function if and only if every royalty holder has expressed acceptance through a transaction to the *retraction* function. IPR and royalty holders' willingness to waive their rights and subsequent negotiations are left to the concerned parties and are not handled by TLSC. Once the current owner successfully calls this function, the token is sent to them, and the Smart Contract is rendered unusable (destroyed or cached, depending on the specific blockchain), as depicted in Figure 38. The specifics of the retraction phase are set during the *Initialization phase*.

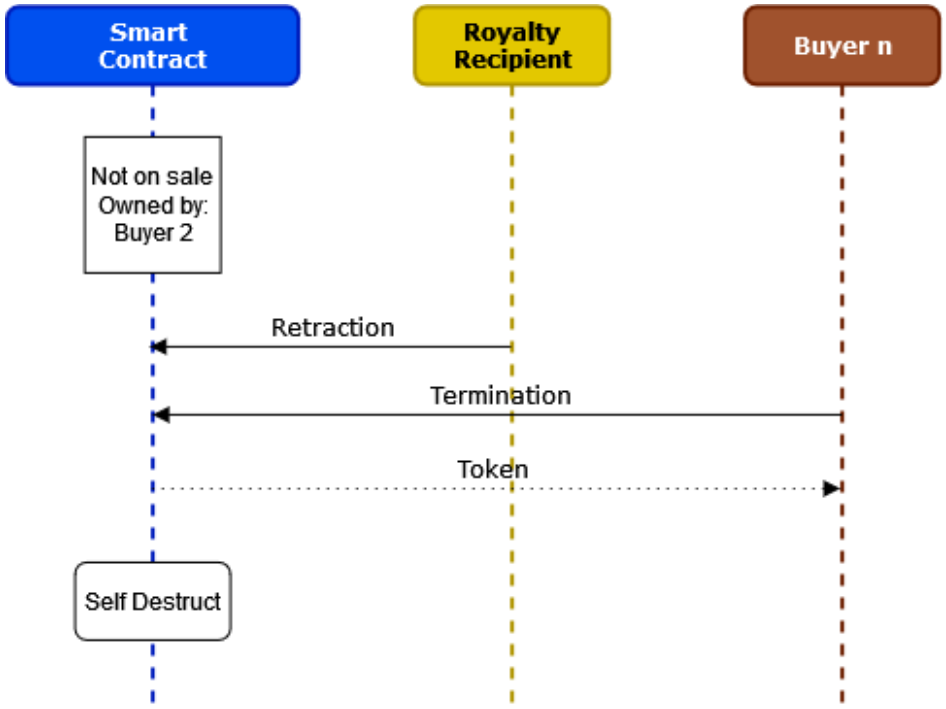

Figure 38: Sequence diagram of the termination phase and end of life cycle of the Royalty Managing Token Level Smart Contract (RM-TLSC).

TABLE III provides a point-by-point comparison of the TLSC's characteristics against the most relevant state-of-the-art equivalents. The five first comparison points deal with features inherited from the design requirements, while the three last deal with supplementary aspects, namely user friendliness and forwards compatibility. Green checks signify compliance, red crosses noncompliance, and orange tildes partial compliance. An implementation of this methodology can be found in Chapter 5. A thorough analysis of the advantages, limitations, and future work relating to the TLSC can be found in Chapter 6.

TABLE III
COMPARATIVE SUMMARY OF THE ADVANCED TOKEN LEVEL SMART CONTRACT SOLUTION AGAINST RELEVANT STATE-OF-THE-ART EQUIVALENTS

	State-of-the-art solutions		**Advanced solution**
	[GTB22a]	[BUR20]	
Flexible rule implementation	✗	✗	✓
Marketplace agnostic	✗	✓	✓
Indeterminate support	✗	✓	✓
Enforcement	≈	✗	✓
Standard tokens	✓	✓	✓
Charges fees	✓	✗	✗
Zero-trust	✗	≈	✓
Forwards compatible	≈	✓	✓

Methodological Solution: The TLSC is a solution that brings a Smart Contract at the level of a token to act as a flexible, zero-trust, indeterminate enforcer of IPR throughout sales of the asset.

4.III. Advanced Smart Contract-centric solutions

4.III.A. Automatic Smart Contract generation

This section intervenes in the same MPEG-IoMT context as Section II.A. As such, the context of ISO/IEC 23093, now MPEG-IoMT, providing architectures, APIs, and data representations similarly applies. Yet, this section deals with another of its aspects: the automatic generation of Smart Contracts representing and executing IPR agreements.

Indeed, past the standard blockchain pain points mentioned throughout Chapters 2 and 3 (computational limitations, memory caps, resource consumption, *etc.*), the use of these specified blockchain integrations still requires high-abstraction operations (*e.g.,* Smart Contract development) that must be provided on a case-by-case basis, as the process is *a priori* blockchain specific. As each blockchain operates with particular programming languages and constraints, the software development process becomes complex and casts doubts about the possibility of specifying a unitary solution and integrating IoMT in blockchain environments. As an incremental step towards the symbiosis of these paradigms, [CHO18] provided the insight that Smart Contracts could be generated automatically from domain-specific ontologies.

We extended the idea and bring forth a solution for automatic IoMT Smart Contract generation and blockchain-agnostic process. The principle consists of automating the production of Smart Contracts around IoMT specifications while allowing fine-grade control and tracking device performed actions, from their activation to the transmission of generated content. The advanced workflow enables media devices and their content to be seamlessly protected by blockchain applicative technology and can be seamlessly integrated with any other complementary protection solution.

Our workflow is supported by a generic, blockchain-agnostic architecture illustrated in Figure 39. It combines expertise from IoMT and blockchain experts and comprises three main components: the IoMT Parser, Smart Contract Developer, and Blockchain Manager. These components are interfaced with IoMT/blockchain experts and MPEG-IoMT APIs.

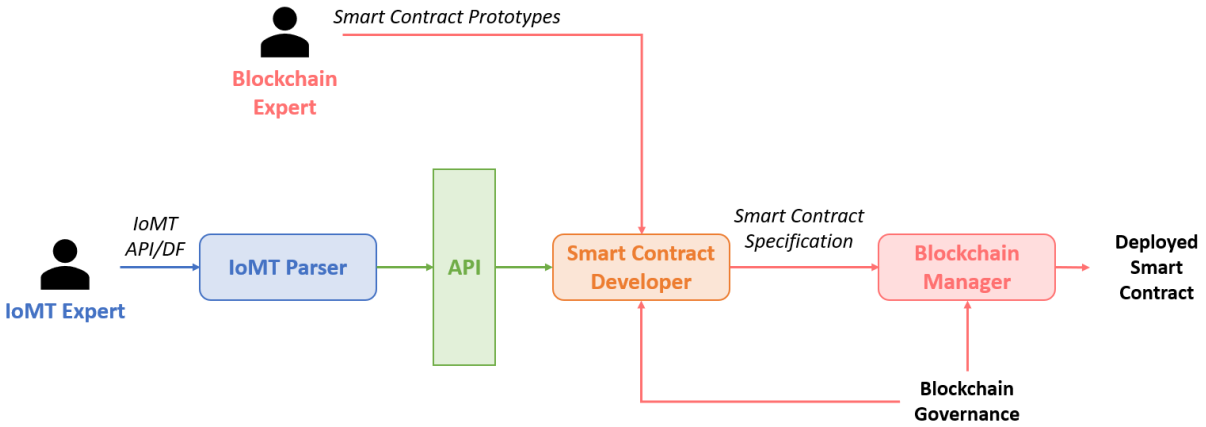

Figure 39: Workflow overview for the automatic conversion of Internet of Media Thing (IoMT) data formats [ISO22b] into Smart Contracts.

IoMT Parser

The IoMT Parser matches information found in XML/RDF documents to structured sets of information data that are formatted according to MPEG-21 standards. Importantly, these formats are blockchain-agnostic.

Smart Contract Developer

The Smart Contract Developer combines Smart Contract Prototypes (or templates) provided by a Blockchain expert and specificities of the blockchain environment (*i.e.,* blockchain governance) and applies them to the formatted data given by the IoMT Parser to produce Smart Contract Specifications. It does so by exploring its input information and matching clauses with functionalities available in its accessed templates. As such, all the IoMT information is represented through accounts, clauses, *etc*. This is, again, blockchain agnostic.

Blockchain Manager

The Blockchain Manager implements the Smart Contract Specification to the specific blockchain in context. It deploys appropriate contracts that are necessary to execute the media contracts on-chain.

The process starts with an IoMT expert who deploys an IoMT system in which at least one blockchain is required to operate at least one MThing. Hence, the IoMT expert designs and develops the software application based on the blockchain-specific IoMT APIs and data formats [ISO22b], which clarify the obligations of the MThing. Yet the IoMT expert is unaware of the technical complexity of blockchain operations. The IoMT Parser extracts the relevant specifications for a blockchain operation and returns a structured set of transactional rules and conditions. This module is independent of the blockchain, and its output can be accessed through a generic API. In parallel, the Blockchain Expert (with deep skills in blockchain technology but limited knowledge regarding IoMT standards) elaborates a set of specific Smart Contract Prototypes for IoMT usage. Although these Smart Contract Prototypes are logically independent with respect to the blockchain solution, their code depends on the specific programming language supported by the blockchain in context. By combining IoMT and blockchain information (Blockchain Governance and Smart Contract prototypes), the Smart Contract Developer produces the Smart Contract specification. This process ensures that the IoMT rules are bound to Smart Contract functionalities. Note that the Smart Contract Developer has both generic and blockchain-specific input information thanks to the API on the one hand and Blockchain Governance and Smart Contract prototypes on the other. Yet, its specification and implementation are blockchain agnostic. This notably opens the door to transferring verified contractual data from one blockchain to another. The Blockchain Manager is the native module of the specific blockchain and is considered the exit point of our architecture as it deploys the Smart Contract for media. This Smart Contract will apply the rules it was setup with indeterminately with no exterior influence, as dictated by **SCDR2**, and compatibly with current standards, as per **SCDR3**. This Smart Contract will store data and potentially manage assets (typically cryptocurrency), as it is hence compatible with **SCDR1** and **SCDR4**.

This workflow demonstrates that specified rules can be automatically implemented as Smart Contracts and be executed for predefined tasks in a limited time despite conventional blockchain solutions' memory, computational, and energy constraints. Its main benefit resides at the operational level: once the architecture is set and deployed, it can serve applicative needs without the implication of Smart Contract developers. The Smart Contract is automatically generated and deployed for each new IoMT-compatible workflow. This leverages high-added-value business models for the realms of IoMT and blockchain. Moreover, the advanced solution is positioned at the IoMT device level and can be integrated seamlessly with any other high-level content protection and security application, such as traditional security solutions (firewalls, authentication tokens, certificates, watermarking, fingerprinting, *etc.*) or novel blockchain-based solutions. These advantages result from the design requirements the solution was built around.

Table IV provides a point-by-point comparison of this methodology regarding its most relevant state-of-the-art counterparts. The first two deal with architectural features, the following two with the expandability of the methodology, and the last with the most expansive feature of our proposed model: the ability to simultaneously manage Smart Contracts and Tokens. Green checks signify compliance, red crosses noncompliance, and orange tildes partial compliance. Two implementations of this architecture are shown in Chapter 5, and Chapter 6 provides an in-depth look at the main advantages, limitations, and further efforts associated with the approach.

TABLE IV
COMPARATIVE SUMMARY OF THE ADVANCED AUTOMATIC SMART CONTRACT GENERATION SOLUTION AGAINST RELEVANT STATE-OF-THE-ART EQUIVALENTS

	State-of-the-art solutions			**Advanced solution**
	[ZUP20]	[FRA16]	[CHO18]	
Human readable inputs	☒	✓	✓	✓
Fully automatic generation	≈	≈	✓	✓
Workflow generality	✓	✓	☒	✓
Cross-blockchain support	≈	≈	≈	✓
Technical compartmentalization	☒	☒	☒	✓
Asset management	☒	☒	☒	✓

Methodological Solution: This workflow enables the systematic generation of Smart Contracts from off-chain ontologies.

4.III.B. Smart Contract load balancing

Hereafter, we focus on the authentication of digital assets using visual fingerprinting, thus allowing for visual content to be tracked via its semantic features. As detailed in Chapter 3, 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, a mainstay in web3 environments.

Coupling blockchains with visual content fingerprinting presents neither conceptual nor theoretical contradictions. Yet, the association between the two is drastically restricted by the lack of methodological bridges and key 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-chain. 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 being created on-chain to make the most out of modern visual content processing in web3-enabled solutions. We take inspiration from Oracles and zk-rollups by moving computation on-chain, as discussed in Chapter 2. The subsequent challenge lies in retaining blockchain advantages, namely immutability. This solution will use a combination of cryptographic hashing and visual fingerprinting, introduced in Chapter 3. Cryptographic hashing will allow us to turn large fingerprints into smaller digests, which are easier to store, while fingerprinting technology will power the near-duplicate content detection aspect of the process.

To this end, we build on and extend ideas brought forth in [ALL21], [ALL21a], and [MOR23a] 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, and (3) Smart Contracts for immutable storage.

Using these tools, semantic features of database entries can be blockchain authenticated and serve to detect near copies in a zero-trust fashion. In achieving the above, this solution contributes an architecture enabling the systematic verification of perceptual features in a blockchain environment featuring (1) a mechanism establishing mutually beneficial

associations of on and off-chain applications for said authentication; (2) an easy-to-operate automated workflow, with built-in protection against mistakes; (3) backward and forwards interoperability with contingent state-of-the-art solutions.

A generic architecture illustrated in Figure 40 supports the processing workflow we advance. It is designed to ensure the processing and exchange of data amongst three logical entities: an off-chain App, a Smart Contract, and an off-chain Database:

- 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.
- The *Smart Contract*: The *on-chain* code is an unfalsifiable integrity check for the *off-chain database*.
- The *off-chain database(s)*: The lightweight databases holding the fingerprints for the recorded content.

The initial setup of the database and deployment of the Smart Contract is done by a qualified blockchain expert. Once setup, no more blockchain expertise is required, and an App operator can use the architecture.

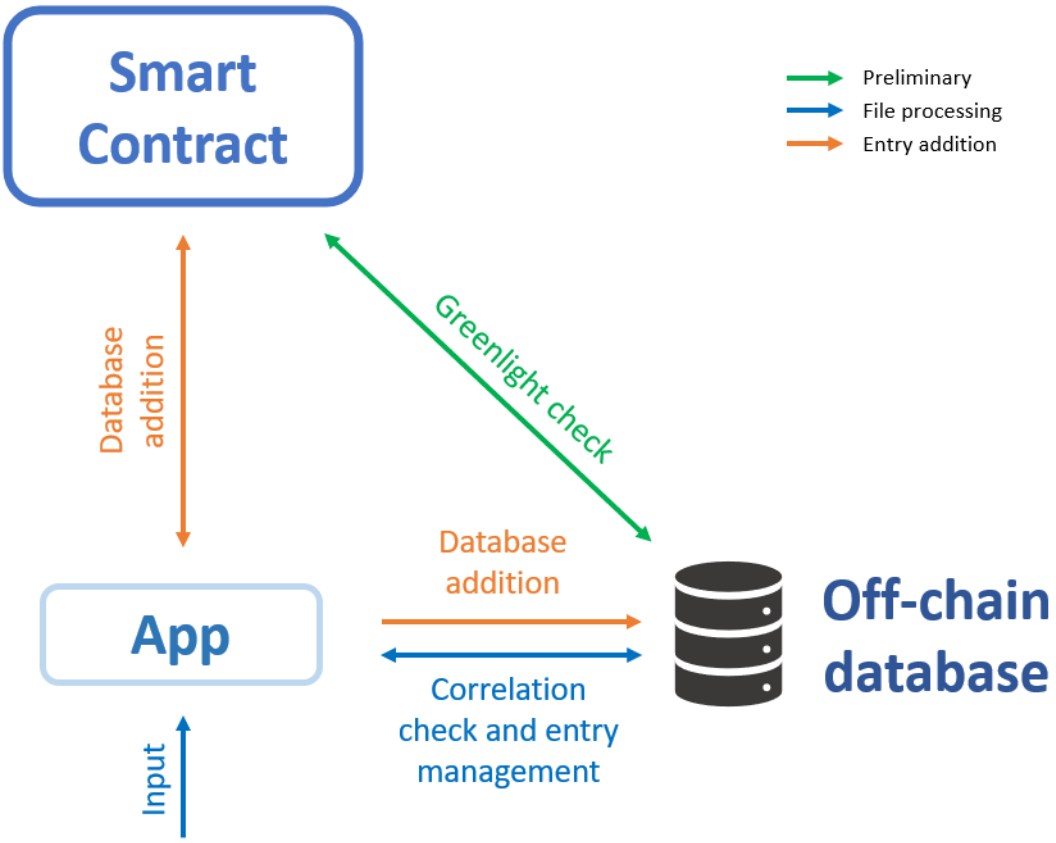

Figure 40: Advanced load balancing architecture, bearing an on-chain Smart Contract and an off-chain App and database.

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 further in Figure 41, can lead to three results:

- 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 near-duplicated content or a *near copy* of one of the entries (*i.e.,* although the digital representation is different, the semantic content is similar to an entry of the database).
- The content is not detected as a copy of the existing entry. The operator can add it to the database by answering a prompt.

If the input is considered original in semantic terms (according to the matching criterion of the fingerprinting method), 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 (through a "greenlight" function detailed hereafter). Once the App has received the confirmation of this operation, it adds the original input to the *off-chain* database.

The database

The advanced architecture does not worry itself with the exact technology managing the database. In fact, it must only bear light lifting, as it only needs to hold the fingerprints of the multimedia content and pass that information to the App when requested. Although it would be possible to hold the content in the database and fingerprint it upon retrieval, a lighter and more private database allows for faster processing and fewer potential privacy concerns.

The Smart Contract

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 further) and to process a new entry admissible in the database. The former does not require input data; the latter requires a hash and an optional string of general information recorded with the entry, which are mapped to a Boolean, indicating their existence. It maps these two entities into a structure containing a Boolean to indicate the existence of the hash and an optional string containing general information. In addition, it implements five functions.

Three of these functions are of *get* type and allow communicating information about the on-chain Database to the App. They return the size of the map, the data associated with a hash, and the Boolean associated with a hash, respectively. The latter serves as the comparison 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, indexes relevant information (if present in the parameters), adjusts the size of the map, and returns a Boolean to indicate successful processing. The deletion function checks for the entry's existence and adjusts the map's size if needed before returning a Boolean. 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. These functions are very simple, and their functionalities can be replicated across all applicative blockchains, making the methodology follow **SCDR1** and **SCDR4**. The Smart Contract also makes no provisions preventing standard compatibility, making it comply with **SCDR3**.

The App

The App has a central role in the process. Not only does it interact with both the database and the Smart Contract (and is the only link to web2 for it, as dictated by **SCDR2**), but it also acts as the only point of contact for the operator. The App is given a multimedia file (whose format is dictated by the fingerprint in context) and 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 Smart Contract's `compare()` function. 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 illustrated in Figure 41.

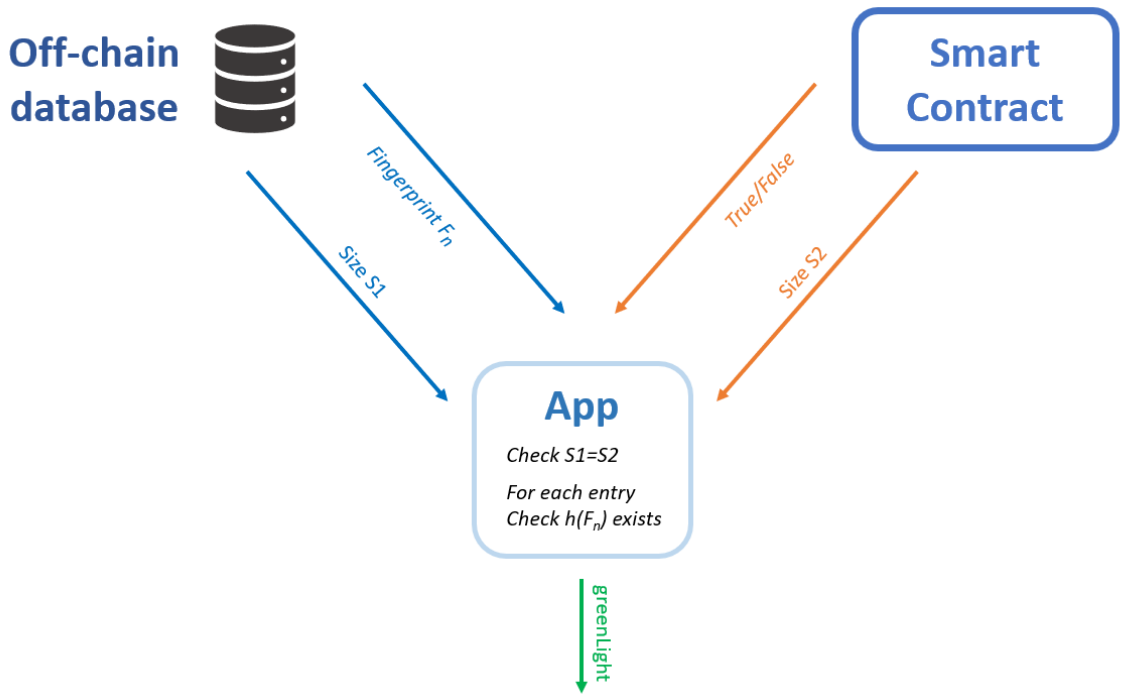

Figure 41: Illustration of the `greenlight()` function's database verification, which unlocks the rest of the data authentication process.

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 using a threshold decided for the use case.

Because of the previous `greenLight()` check, this processing can be performed entirely *off-chain*, enabling this workflow's computational efficiency. If semantic similarities are found between the input and at least one of the entries, it may not be introduced to the database. Before informing the operator as such, the App determines whether it is dealing with a strict or near copy using a fast check using the recorded cryptographic hashes. Some leeway could be given to the near-copy case, where an operator could be given the discretion to validate an entry detected as a near-copy. This is left to the specifics of implementation and use cases (cf. Chapter 6). This process is illustrated in Figure 42.

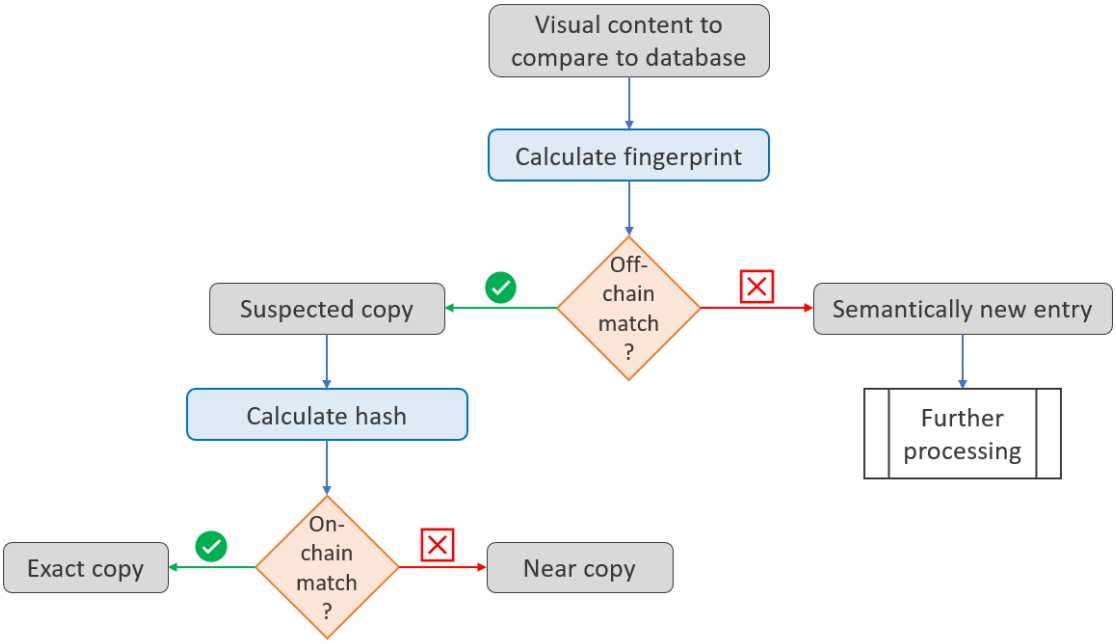

Figure 42: Process diagram for the semantic content comparison of visual media input in the advanced authentication process.

If the input is shown to be semantically original enough, 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 [ISC23a] considers four different 72-bit strings), the hashing step may be skipped as it is not essential to the proper 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 43.

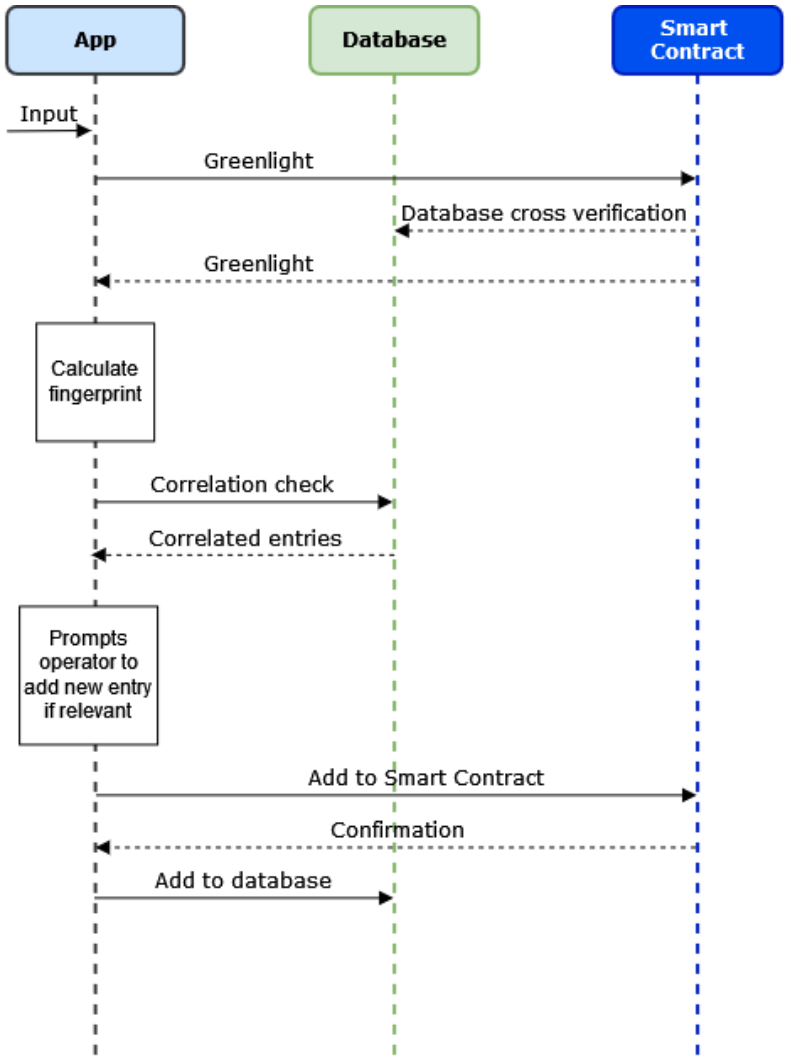

Figure 43: Sequence diagram for the addition of a new entry in the blockchain-backed database.

Although the method used to identify content (*i.e.,* the fingerprinting method) is the application's core, the general architecture is independent of its specificities. The role of the fingerprinting method is twofold. First, being the initial step of the process, it defines the input format. Indeed, near copy detection has use cases using a variety of data formats (images, video, text, *etc*.), some of which might focus on semantic content, while others could include metadata or instance data. Second, the detection can only be as precise as the specific fingerprinting method permits. Rather than having a universal solution, appropriately selecting a fingerprinting method on a case-by-case basis will yield the best results (cf. Chapter 6). The thresholds paired with the fingerprinting methods used to detect near copies depend on the use case. If the objective is only to detect very close copies of the content in the database, we would set our normalized correlation threshold close to 1 or our maximum Hamming distance very small (in the range of 0 – 3 bits for a 72-bit identifier). If we are more generally looking to detect the same semantic content after alteration, we would set our normalized correlation threshold between .6 and .8 or our maximum Hamming distance between 8 and 12 (for binary fingerprints of size for a 72-bit identifier).

The architecture and framework provided above enable multimedia content tracking to be backed by blockchains. We use a load balancing architecture to enable the complex computation to be possible in such environments and provide a mutually beneficial relationship between the applicative bricks available on-chain and off-chain. Indeed, blockchains bring trust and immutability to advanced content detection, while multimedia processing brings a level of perceptual feature detection that cannot be accomplished natively on blockchains. Moreover, most is made of minimal blockchain processing, which is slow and power-hungry. This method also makes the most of the flexibility of its components to host state-of-the-art fingerprinting techniques, never restricting their features and performances, which makes its spectrum of potential uses extensive. Additionally, the methodology can seamlessly be exported to various blockchain environments and use multiple database technologies, as only the most widely available functionalities are required. The data verified thusly can be used in many contexts, including ones that need robust Intellectual Property features.

Table V provides a point-by-point comparison of this methodology against its most relevant state-of-the-art counterparts. The first three criteria deal with the architectural design of the methods, the following two with adaptability concerns, and the last with the main advantage of the advanced solution, namely the capacity to distinguish near copies. Green checks signify compliance, red crosses noncompliance, and orange tildes partial compliance. An implementation of this architecture is shown in Chapter 5, and Chapter 6 provides an in-depth look at its main advantages, limitations, and future work.

TABLE V
COMPARATIVE SUMMARY OF THE ADVANCED SMART CONTRACT LOAD BALANCING SOLUTION AGAINST RELEVANT STATE-OF-THE-ART EQUIVALENTS

	State-of-the-art solutions		**Advanced solution**
	[ALL21a]	[ALL21]	
Computation between environments	✅	❌	✅
Load balancing architecture	❌	✅	✅
Zero trust compatible architecture	≈	✅	✅
Robust against database tampering	❌	❌	✅
Flexible implementation	❌	≈	✅
Near-duplicated content detection	❌	❌	✅

Methodological Solution: This load balancing architecture allows to process visual content off-chain in a blockchain-authenticated fashion.

4.IV. Advanced integrated solutions

4.IV.A. Foundations and direct applications

The methodologies in Section II and Section III answer given sets of circumstances dictated by industry necessities and gaps in the state-of-the-art identified in Chapter 3. Furthermore, they were designated not only to answer interoperability questions regarding blockchains (within blockchains, between blockchains, and between blockchains and web2 computing) but as interoperable entities themselves. As such, their combination is possible and can enable further features that contribute to the above use cases. In fact, some of the technologies presented were developed congruently. The methodologies we will show throughout this section will answer the relevant requirements introduced in Section I.

Notably, the IoMT-related methodological bricks, namely the IoMT Broker (II.A) and IoMT automatic Smart Contract generation (III.A.), were developed as one. Their association was published in [ALL21a]. With the details provided earlier in this Chapter, the association context is elementary: Smart Contracts representing MThings are generated automatically and interact with the IoMT broker to make the MThing's data available for purchase in a systematic and zero-trust fashion. Figure 44 shows this interoperability by combining Figure 33 and Figure 39. Chapter 6 will show an implementation of this integrated architecture and workflow rather than illustrating them separately. In this case, compliance with SCDRs and TDRs is directly inherited from the separate methodologies, as no changes are made to the generated Smart Contracts and to token management. Similarly, tokens meant for mass distribution can be equipped with a TLSC (cf. Section II.B.) to enforce rules, limitations, or royalty payments.

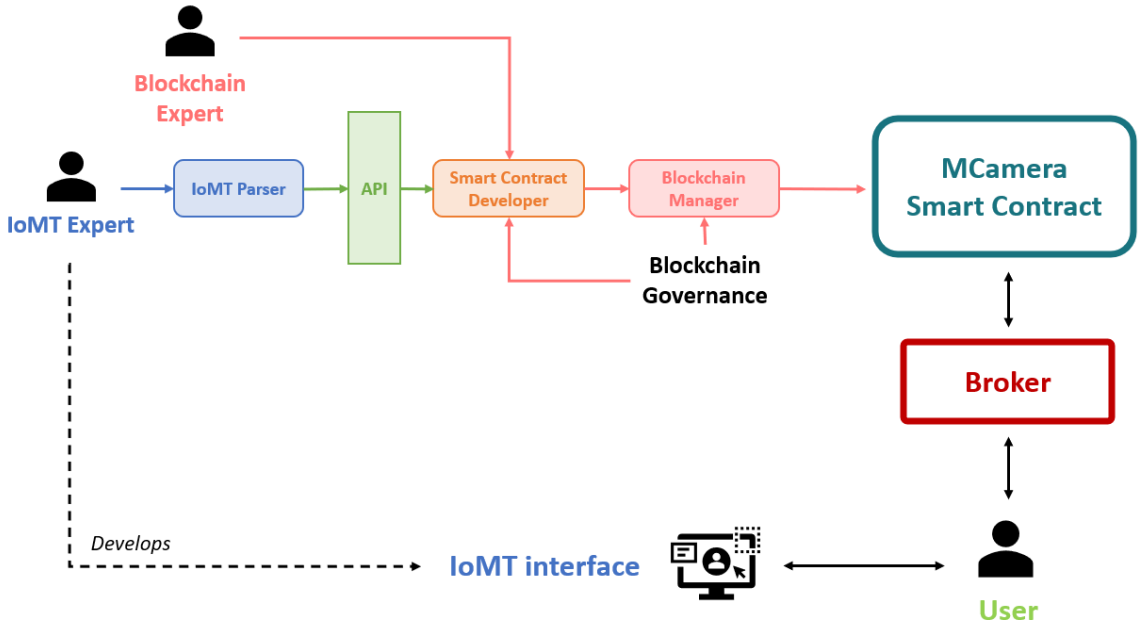

Figure 44: Overview of the combinatory Smart Contract generator and broker that enables the automatic distribution of Internet of Media Things (IoMT) content.

The rest of the Section will be dedicated to more complex technological combinations, namely a complete MPEG IoMT infrastructure standardized in a White paper and an architecture managing the entire lifecycle of visual assets in blockchain environments, which will make the most out of the previous innovations.

4.IV.B. Media Smart Contracts

This architecture was built upon in the context of ISO/IEC JTC1 SC29 WG3, where we collaborated and implemented a larger framework based on the same technical approach as the Smart Contract generation put forth in Section III.A. This effort originated targeting issues with overlaps to the ones dealt in our royalty-enabling token solution (Section I.B), namely IPR and royalty distribution issues in the multimedia industry, tackling the "black box" issue. Specifically, the use case resides in automatically translating media contracts from the music industry into Smart Contracts that encode music and media asset trading terms and conditions and can execute their clauses automatically and transparently. This solution provides the other approach of royalty distributions on blockchains to complement our TLSC, namely Smart Contract-based solutions. The base human-readable contracts identify parties (*e.g.,* licensor, licensee, *etc*.) and their obligations (*e.g.,* income splits, licensing agreements, *etc*.) but are set in specific standardized formats before being put up for the translation process, as per **TDR1**.

Yet, publishing the Smart Contract does not ensure that the clauses of the narrative and Smart Contract correspond. Thus, an important feature is the possibility to bind, through persistent links, Smart Contract clauses to their corresponding ones in the narrative contract and vice versa; *e.g.,* the narrative clause "*user A pays $1 to user B*" is bound to its counterpart SC clause "*Transfer UserA UserB $1*". To accomplish this, the main contribution of this standard is a method comprised of an architecture and a workflow enabling the bidirectional conversion between MPEG-21 CEL/MCO contracts to/from Smart Contracts. Hereinafter, we describe the forward conversion. The backward conversion follows the same process in the opposite fashion. Both are illustrated in Figure 45, which presents numerous similarities with Figure 39.

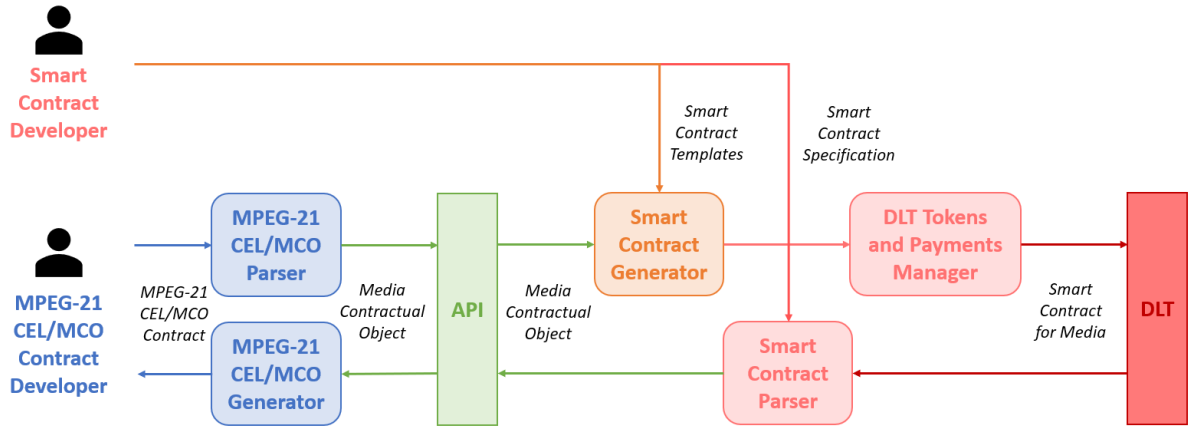

Figure 45: MPEG-21 Contract Expression Language (CEL)/Media Contractual Object (MCO) contracts to and from Smart Contracts conversion workflow.

Briefly, The *MPEG-21 CEL/MCO Parser* is a component that gets as input an MPEG-21 CEL/MCO contract and produces as output a set of blockchain-agnostic MCOs. The *Smart Contract Generator* gets a set of MCOs and DLT-specific Smart Contract templates as inputs and produces a blockchain-specific Smart Contract specification, *i.e.,* elements representing the information needed to deploy the SC. Finally, the *DLT Tokens and Payments Manager* receives a Smart Contract specification, a set of blockchain addresses representing contract parties, and a DLT governance protocol, *i.e.,* the rules used by the blockchain to update the ledger while producing a Smart Contract, which is deployed on the given blockchain. Although the process of generating the Smart Contract is complex, the resulting code simply executes contractual stipulations through data storage and asset management, ensuring **SCDR1**, **SCDR3**, **SCDR4**, **TDR2,** and **TDR5**. They do so automatically with no exterior influence, making the overall workflow compatible with **SCDR2, TDR4,** and **TDR3**.

This architecture carries the advantages presented in Section III.A and is the application of the idea to the field of Media Contracts. As such, we shall provide a joint implementation and analysis in Chapters 5 and 6, respectively.

Intermediate Methodological Solution: The combination of our automatic Smart Contract generation and brokerage solutions can provide the systematic and transparent execution of contractual clauses.

4.IV.C. Asset lifecycle management for visual content tracking

This solution tackles the issue of semantic content detection in blockchain environments we introduced in Section II.B, albeit in a much more thorough fashion. It expands the idea to feature the automatic tokenization of perceptual features and their subsequent protection and distribution, as well as the support of simultaneous token standards and blockchains. With these features, we create and automate a systematic process for managing decentralized assets representing physical or web2 assets being created *on-chain* to make the most out of modern visual content processing. These assets can then be upgraded with blockchain protection functionalities, handed to state-of-the-art on-chain brokers, and be 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.

Figure 46 illustrates the main successive processing steps assets undergo during our process. The specifics of this workflow are detailed hereafter.

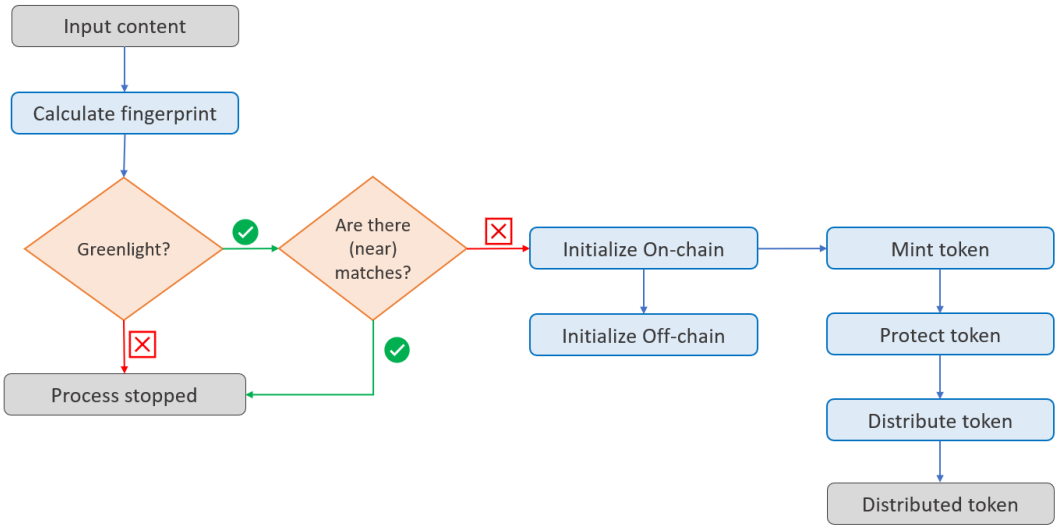

Figure 46: High-level workflow of our full lifecycle token management solution, starting with a piece of visual content and ending with a distributed token.

The workflow illustrated above and detailed hereafter presents similarities with the load balancing method presented in Section III.B. in its initial phases, represented by the steps prior to "Mint token" in the above. We shall restate the workflow but position it relative to the design requirement philosophy preceding the figure.

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. This content already has stipulations on its IP and licenses established in either the physical or web2 worlds, in alignment with **TDR1**. 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 detailed in Section II.A. Simply storing the fingerprints would make them vulnerable and static, as they could not be used to the full extent of content distribution. Consequently, we initialize this information on the blockchain to benefit from

immutable data tracing native to the environment and comply with **TDR3** before storing the fingerprints. Storage resource limitations naturally invite us to hash the information before recording it *on-chain*. Once certain the information appears in a Smart Contract, we store the initial fingerprints in an *off-chain* data storage. This redundancy will enable advanced checks we explain in depth 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. The Smart Contract that manages these operations inherits all SCDR compliance from the underlying methodology in Section III.

The second portion of the workflow, illustrated by the rightmost column of Figure 46, uses the token-oriented solution presented in Section II of this chapter.

At this point of the process, we have an immutable trace of semantic data processing that establishes 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 **TDR5**), we create a standardized token (as presented in Section II.B.2), which includes links to the data storage entries. The fact that this token is minted with information that the Smart Contract verifies ensures the systematic appreciation of underlying information brought forth in TDR1 and makes this process follow the logic dictated by TDR4. Furthermore, this token can either represent the original or a related asset, *e.g.*, an asset representing exploitation rights or IPRs. We then protect these tokens with state-of-the-art solutions (as presented in Section II.B.3) and hand them to a Token Broker for distribution. 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 **TDR5**, **SCDR3**, and **SCDR4**. We designed the self-sufficient (**TDR2**) architecture shown in Figure 47 to support this workflow.

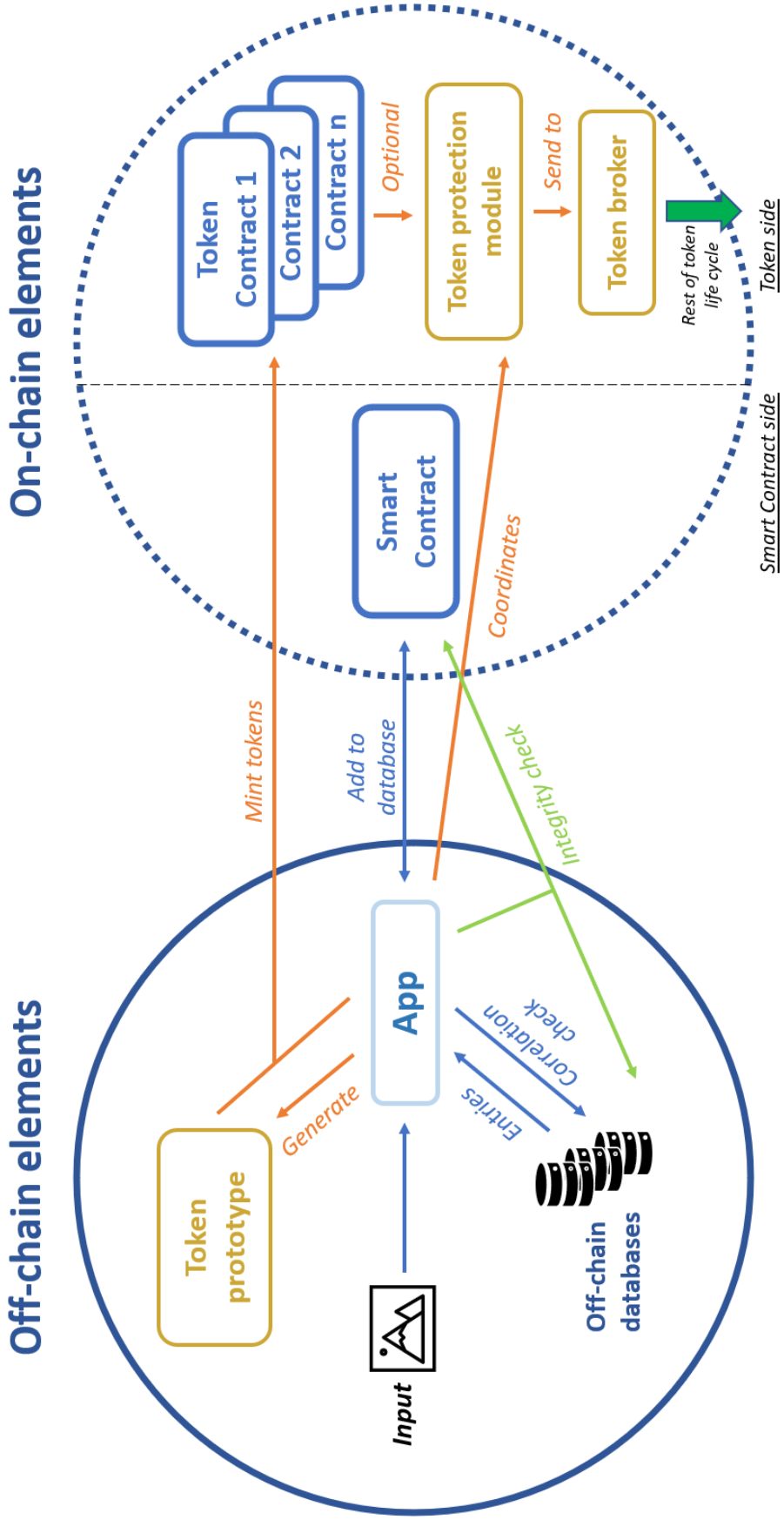

Figure 47: Complete architecture powering the complete lifecycle management of web3 assets connected with web2 visual content.

As with our previous methodologies, a qualified expert sets up the architecture, and subsequent operations can be executed regardless of technical knowledge with built-in protection against mistakes.

The core of the architecture, comprised of the App, Smart Contract, and off-chain databases, constitutes the architecture explained earlier in this manuscript.

The only modification lies in the accommodation of multiple databases. Indeed, use cases might require categorizing content according to format, origin, *etc*. 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. The existence of multiple databases does not impede any functionalities and opens use cases where data can be stored across various data storages.

Once the `greenLight()` function confirms database integrity, the input is detected as semantically original enough and subsequently added to the database and Smart Contract; the input is made into an NFT that represents the perceptual features of the input and the previous authentication process. This NFT's metadata can contain the hash of the input's fingerprint, the transaction number of its inclusion in the Smart contract, or the electronic signature of the operator.

As with databases, many Token Contracts can be used simultaneously across different blockchains. To format the token inputs uniformly, we take a page out of our IoMT solutions (III.A, IV.B) and use a prototype approach. These inputs can then be sent to various Token Contracts deployed across different blockchains. Some blockchains can support a variety of formats out of a single Token Contract (*e.g.*, ERC1155 [RAD18]). In contrast, others will require multiple Token Contracts to be deployed by the expert during the setup phase. Note that the various Token Contracts need not be deployed on the same blockchain nor as the same blockchain as the Smart Contract. Depending on the use case, these tokens can automatically be sent to the deployer wallet, IPR holders, or marketplaces.

The workflow also makes provisions for the use of data protection solutions for the token. Within the context of this thesis, we naturally offer the use of a TLSC (II.B), which can enforce a set of rules transmitted by the app indefinitely.

Here is a summary of consecutive steps taken in the processing of an input, further illustrated in Figure 48. 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 with said inputs and mints a unique NFT.
- *Step 7:* The NFT is empowered with the required security resources and passed on to the desired Token Broker.

This workflow's main strengths lie in the full lifecycle support of assets whose content is controlled before they are minted up to their distribution and in the flexibility and scalability of the approach. The flexibility is inherited from the original load balancing architecture, where every component can be swapped for the most appropriate methodology without de-naturing the process, enhanced by the creation of assets that can be used within or alongside a wide variety of contingent solutions at the service of a broad spectrum of use cases. The scalability comes from the minimal impact of technical expansions on computational resources and gas costs. Indeed, the Token Contract is the only architectural brick that requires multiplication when the workflow is put at the service of multiple input formats, token standards, and blockchain infrastructures.

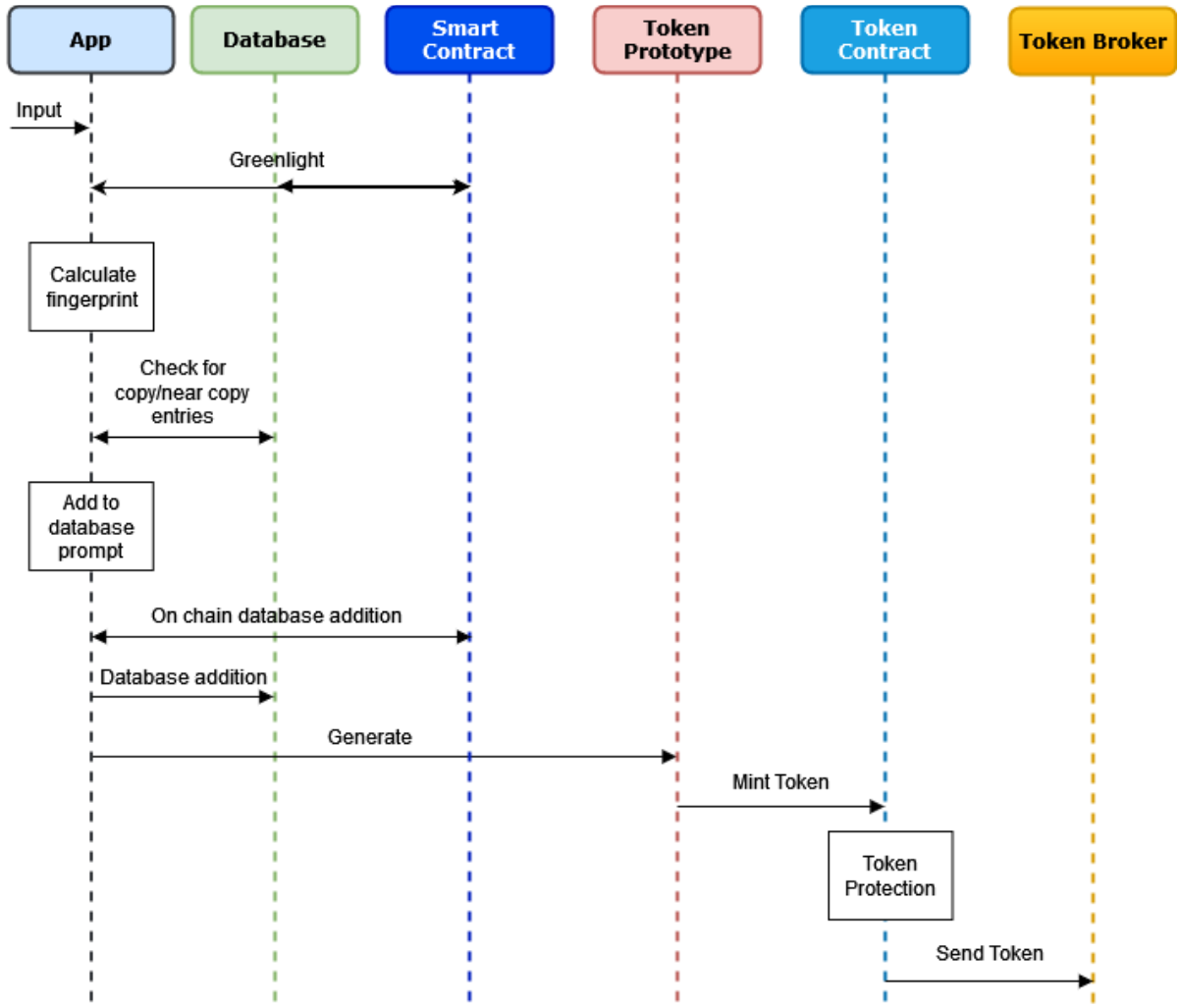

Figure 48: Sequence diagram of the addition of a new piece of content to the authenticated database and the subsequent asset creation, protection, and distribution.

In Table VI, we provide a point-by-point comparison of the features supported by our approach against the closest state-of-the-art solutions. The first three characteristics deal with functional aspects related to visual content tracking, the two following with applicative blockchain concerns, and the last four with horizontal features. Green checks signify compliance, red crosses noncompliance, and orange tildes partial compliance. Two implementations of this architecture at the service of real-world use cases are shown in Chapter 5. Chapter 6 provides an in-depth look at the main advantages and limitations of the workflow and identifies avenues for future work.

TABLE VI
COMPARATIVE SUMMARY OF THE ADVANCED FULL LIFECYCLE BLOCKCHAIN ASSET MANAGEMENT SOLUTION AGAINST RELEVANT STATE-OF-THE-ART EQUIVALENTS

	State-of-the-art solutions			**Advanced solution**
	[MOR23a]	[ALL21a]	[ALL21b]	
Near-copy detection	✓	✗	✗	✓
Load balancing architecture	✓	✗	✓	✓
Blockchain-backed storage	✓	✗	✓	✓
Asset Tokenization	≈	≈	✗	✓
Token distribution	✗	✗	✗	✓
Simultaneous standards	≈	≈	✗	✓
Simultaneous blockchains	✗	✗	✗	✓
Automated workflow	✗	✗	✗	✓
Full lifecycle support	✗	✗	✗	✓

Methodological Solution: The combination of our four base methodologies can provide a full lifecycle management of blockchain assets representing visual media.

Chapter 5. Implementations and analysis

This chapter provides Ethereum and Tezos open-source implementations of the methodologies introduced in Chapter 4 and apply them to relevant use cases, showcasing their contribution to the applicable interoperability front. Then, we will combine all architectural components to offer a complete lifecycle solution for blockchain assets, demonstrating the ability of our technologies to be integrated into larger frameworks. For each of these implementations, we will provide an in-depth analysis of our approach and its performance from a conceptual and practical point of view. We will highlight the most valuable aspects and most significant limiting factors of these solutions, putting them in perspective with the current landscape of web3 visual content and highlighting potential avenues for future work.

5.I. Generalities

In this section, we provide illustrative implementations and showcase the process, results, and analysis of the architectures and workflows presented in Chapter 4. Specifically, we will first showcase a combination of IoMT automatic Smart Contract generation and Brokerage system in Section I, as they were developed in unison. Then, we showcase our Token Level Smart Contract solution in the context of royalty management in Section II. Third, we will showcase how our load balancing architecture can support visual fingerprinting in Section III. These core methodologies will be explained thoroughly by describing their software components and workflow. Finally, Section IV will showcase what the combination of these approaches can accomplish, illustrating its use for two real-world use cases. The first will deal with visual content IPR from the perspective of a museum, and the second will provide full-lifecycle MThing IP asset management. These implementations and illustrations were developed in two different environments:

- We use Tezos to illustrate our solutions' compatibility with less populated infrastructures and to make the most of SmartPy, an online Tezos IDE available through a Python library. SmartPy provides us with clear test scenario capabilities, allowing us to illustrate the use of the Smart Contract to readers unfamiliar with Smart Contract development. As a reminder, Tezos development is based on meta-programming; as such, the code we write is not directly run but serves to construct the actual Smart Contract that will run on the blockchain [TEZ23c]. Table VII summarizes the environment used for our Smart Contract demonstrations.

TABLE VII
GENERAL PROPERTIES OF THE ENVIRONMENT USED FOR OUR TEZOS IMPLEMENTATIONS

Blockchain	Tezos
Environment	SmartPy (legacy)
Language and version	SmartPy v0.16.0
Token standard	FA2

- EVM environments will support the full implementations of our solutions. While some Smart Contracts and transactions will be illustrated through the Remix IDE, as the interface provides better representation for what we attempt to showcase, the architectures are deployed on a 3-node, Hyperledger Besu EEA (Enterprise Ethereum Alliance [EEA23])-compliant PoA private blockchain deployed on an Amazon Web Services server, as well as on the now deprecated Rinkeby Ethereum testnet accessed through the Infura node cluster. An Alethio Lite Explorer [GTB22b] is used to show blocks and transactions. We interacted with the deployed Smart Contracts using the web3py library. Table VIII summarizes the environment used for complete implementations.

TABLE VIII
GENERAL PROPERTIES OF THE ENVIRONMENT USED FOR OUR ETHEREUM IMPLEMENTATIONS

Blockchain	Ethereum
Environment	Remix VM (Merge), EEA private blockchain
Language and version	Solidity 0.8.18
Token standard	ERC721

5.II. Brokerage and Automatic Smart Contract generation

This section goes over the implementation of our automatic Smart Contract generation solution and how it interacts with IoMT APIs and content delivery solutions and showcases the similar technological bricks that went into MPEG-IoMT SCM reference code, which we contributed to as part of a team of ISO/IEC JTC1 SC29 WG3. Respectively being projects undertaken with another active contributor from the same department (namely M. Allouche) and within a working group, we shall be a bit briefer in this section and allocate more space to projects undertaken as a PhD student/supervisor pair later in this chapter.

5.II.A. IoMT in web3

Figure 49 shows the workflow introduced in Chapter 4, Section III and focuses on the data formats output by the different steps. The first step undertaken in deploying IoMT Smart Contracts is the operation of the IoMT Parser, which formats the input for the IoMT API (ref or annex). This parser extracts relevant specifications and returns a set of transactional rules and conditions formatted in an easily searchable, blockchain-agnostic fashion. Specifically, its output is a JSON file that categorizes essential information such as IP addresses, port numbers, wallet addresses, and initial costs in an organized fashion. The leftmost focus of Figure 49 shows an output example for the MCamera use case we introduced in Chapter 4, Section II. Because this input is used regardless of the eventual deployment blockchain, it contains information regarding all the blockchains the original data makes provisions for.

This information is fed to the API, whose output can be instantiated for a specific blockchain through a Smart Contract Prototype. These prototypes enable the data to enter web3 through the specifics dictated by Blockchain Governance and to resemble a Smart Contract's structure, as highlighted by the center focus of Figure 49. The appropriate functions and variables are selected to represent the data output by the API, specified as fully-fledged functions, and made into a Smart Contract that gets deployed on the blockchain. The last focus of Figure 49 illustrates the exploration of an Ethereum Smart Contract deployed as such.

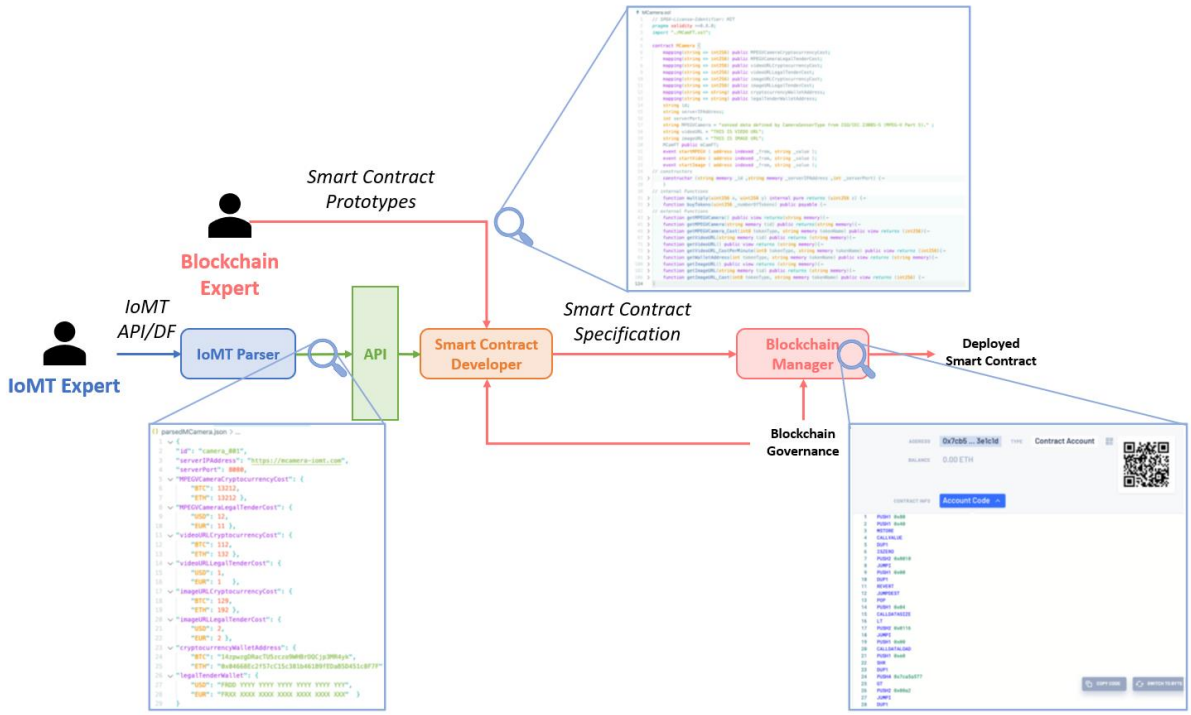

Figure 49: Workflow overview for the automatic conversion of Internet of Media Thing (IoMT) data formats into Smart Contracts, showing extracts of the files generated at each step, namely a parser output, a Smart Contract prototype, and an explored Smart Contract, respectively.

Although the deployed IoMT Smart Contract represents the endpoint of this architecture, it is only the starting point of operation. The IoMT Smart Contract connects to a broker that handles the distribution of tokens, (cf. Chapter 4, Section I). IoMT standards make the provision for the distribution of the content, and we are left to manage brokerage and escrow. The cornerstone of these functionalities is the function, which is paid and delivers the tokens. This function gets the required prices directly from the Token Contract deployed via the IoMT Smart Contract specifications, checks for the availability of tokens, and transfers the appropriate number of tokens according to the received payment, launching a `purchase()` event that can be caught to trigger further operations. The above is illustrated for a single MCamera. When these tokens are paid back to the Broker, it liberates the funds and sends them to the relevant MThing. The exploration of such a transaction for purchasing eight tokens costing 1ETH each is also shown in Figure 50.

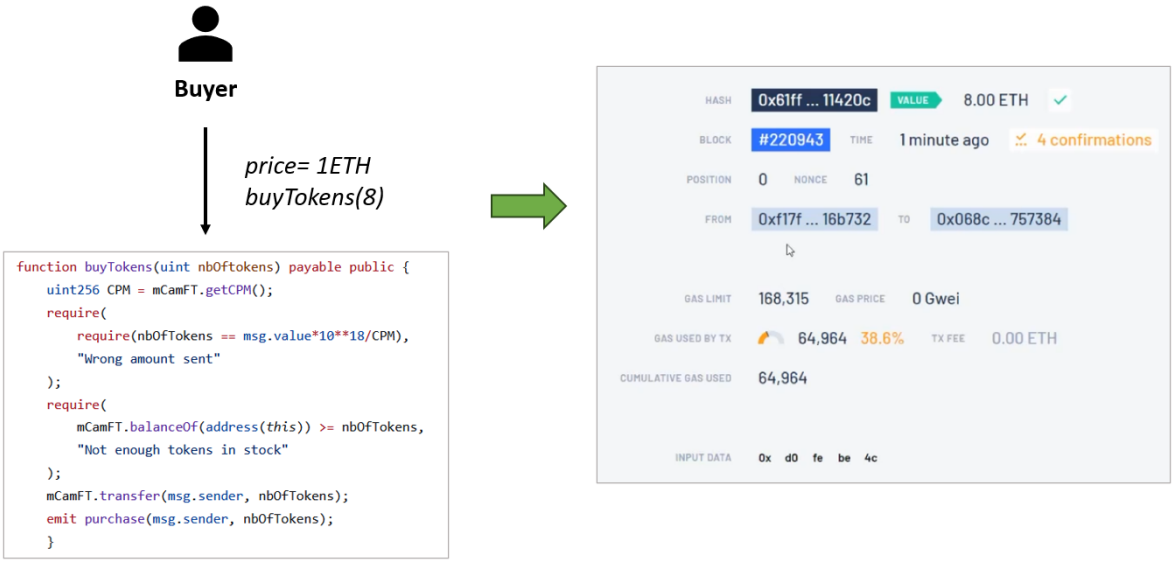

Figure 50: Illustration of a transaction to the `buyTokens()` function (botom left) and resulting transaction observed on our private blockchain (right).

5.II.B. Translating ontologies to Smart Contracts

A similar approach to the one above was used in the MPEG-SCM code we contributed to. Its goal was to create a bi-directional pathway between standardized ontologies and Smart Contracts. The initial input of the architecture is a media industry contract, of which use cases are instantiated according to the Open Music Initiative (OMI) [OMI23], *e.g.,* imitating a streaming deal between an artist and a big music label. These inputs are then parsed into MCOs with a specialized Python code. A portion focused on extracting contractual parties is shown in Figure 51. The resulting MCOs are then used alongside Smart Contract Prototypes to generate a Smart Contract specification. This Smart Contract includes functions that execute contractual obligations, such as paying parties, as shown in Figure 51, and manipulates a variety of Media Tokens representing obligations and IP entities that are eventually deployed alongside it.

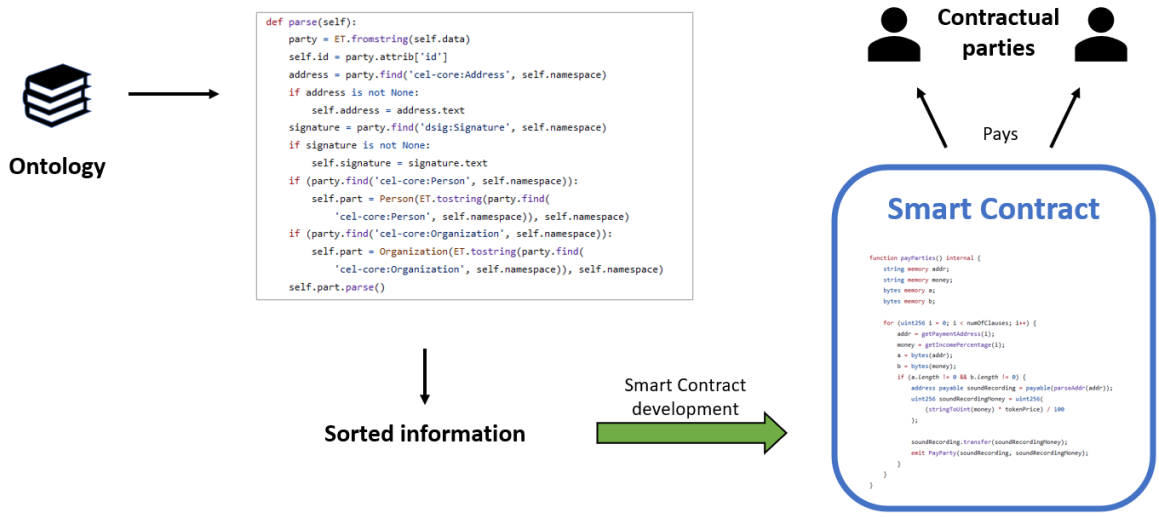

Figure 51: Extract of the parser (left) and the eventually resulting deployed Smart Contract (right).

The deployment constitutes the final step of the forward process. The explored deployment of one of these Smart Contracts is shown in Figure 52, alongside a focus on a Tezos Smart Contract's data from SmartPy's test sandbox environment, specifically for OMI's download-big-label use case.

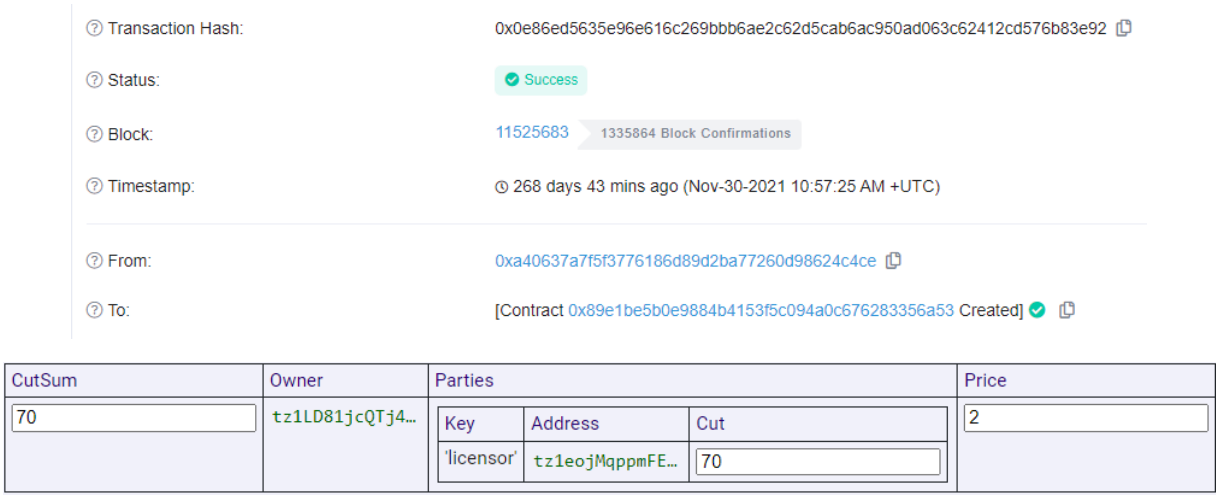

Figure 52: Exploration of a Smart Contract deployed on Tezos using the automatic generator (top) and its storage variables (bottom).

The approach and software also enable a backward conversion, generating CELs from deployed Smart Contracts. The methodological bricks and their execution are the same, and their complete implementation is available alongside the forward conversion provided with this thesis. Although the development process was detailed in Section II.B. using MVCO, the same procedure was followed for AVCO (adjusted to its specificities), and the reference software and demonstration are also provided alongside the MVCO implementation.

5.II.C. Analysis

The results presented in the previous sections establish the concept, design, and realization of IoMT and SCM Smart Contract production. These Smart Contracts can automatically, reliably, and transparently execute predefined tasks already formalized in the context of MPEG-IoMT and MPEG-SCM standards. These simple and strictly defined tasks are naturally adapted to blockchains as they do not suffer from the computational limits of the environment in terms of memory or costs. Standards that are not implemented can be converted into deployed Smart Contract without human intervention using these solutions. This process can be expanded to new blockchain infrastructures, barring the establishment of a single Smart Contract Prototype (stable as it follows appropriate standards), and subsequently support IoMT and Media contracts (respectively) without the need for manual development.

This approach is of high value for business models and operational schemes where requirements are already formalized, and only minor variations allow a base model to span most use cases. This fact is then supported by the interoperability of the workflow, which is compatible with any other high-level content protection or security application, such as traditional security solutions (firewalls, authentication tokens, certificates, watermarking, fingerprinting, *etc*.) or novel blockchain-based solutions.

Further work can be divided into four categories: the expansion of blockchain supported through the development of Smart Contract Prototypes; the specification of an open API for exchanging Blockchain Governance information and enabling further interoperation; the furtherance of features in accordance with new blockchain advancements; and finally, the expansion of possible technological realms. Indeed, any field that uses ontology classifications or similar data structures could implement the same approach.

5.III. Token Level Smart Contract

In this Section, we discuss TLSC implementations. We will go over the decision-making of its specification on Ethereum and a test scenario in Tezos to show the infrastructure flexibility of the solution. As a reminder, the TLSC is based on a single Smart Contract containing a token, handling IP limitations and royalty distributions. Given that the former is more obscure, we shall focus on showing the latter. A TLSC specifically instantiated to manage royalties is referred to as a Royalty Managing Token Level Smart Contract (RM-TLSC). These tests were run with a gas cost of zero. Actual implementations would have the Smart Contract paying for gas, hence keeping the required sums out of the purchases available in its balance. The scenarios we used to run the tests were inspired by the ISO 21000-23 Smart Contracts for Media standard previously discussed throughout Chapter 4 (Section II.A, Section III.A, and Section IV.B).

Our use case entails (1) a *creator*, the original user that initializes the RM-TLSC; (2) *IPholder1*, who, in contributing to the work, secured a 20% royalty share; (3) *fakeIPholder*, who undeservedly tries to gain a royalty share; (4) *randomAddress*, an unnamed party appearing due to an error; and (5) *buyer1* and *buyer2*, purchasers of the RM-TLSC. Please note that because they are not specified as a royalty holder, the *creator* is not entitled to subsequent sales portions and will receive compensation only for the initial sale. This is summarized in Table IX. The 80% remaining share will go to the seller in context.

TABLE IX
SUMMARY AND DESCRIPTION OF THE PARTIES USED IN THE TEST SCENARIOS OF OUR TOKEN LEVEL SMART CONTRACT IMPLEMENTATION

Parties	Description	Share
creator	The initializer of the RM-TLSC.	0%
IPholder1	A royalty shareholder.	20%
fakeIPholder	A malicious party trying to gain an unwarranted share.	0%
randomAddress	An unnamed party appearing due to an error.	0%
buyer1	The first purchaser of the RM-TLSC.	0%
buyer2	The second purchaser of the RM-TLSC.	0%

5.III.A. Implementation

In this example, we use the popular ERC721 NFT standard [ENT18], although any other would function similarly. The complete Smart Contracts are available alongside this manuscript, including a complete Truffle framework, a test NFT, and web3py [WPY23] testing scripts. We also provide the code and further testing on the Remix Integrated Development Environment (IDE). The Remix project contains two executable testing scenarios, one going through the complete token lifecycle successfully, the other highlighting fraudulent behavior not tolerated by the RM-TLSC.

Let us now dive into specifics. The Smart Contract stores three types of variables: the ones relevant to the owner's status, the token, and the royalty information. The former records the original and current owners, checking for the first transfer limit. The second includes the NFT, specific Id, and the currently listed price. The latter notably contains the list of beneficiary addresses, their respective cuts, the total royalty percentage, and the retraction check. These variables are set during the initialization phase, except for the dynamic owner, price, and retraction check. These are illustrated in the scenario explained hereafter in Figure 53.

Figure 53: Owner and token-related variable states of a Token Level Smart Contract (TLSC) after an Ethereum test scenario initialization.

Two modifiers, namely `onSale` and `onlyBy`, will ensure functions are only called by authorized addresses when the owner allows them to. They are shown in Figure 54.

Figure 54 Modifiers in used in the Token Level Smart Contract Ethereum implementation.

This Smart Contract contains eight functions, five of which are generic and three of which are specific.

The basic descriptions of the five generic functions are:

- *constructor*: allows the creation of the Smart Contract. Interestingly, it includes the specific token down to the Id to avoid costly mistakes.
- *changePrice*: allows the current owner to modify the listed price.
- *sale*: allows the current owner to define whether the token is available to purchase.
- *retraction*: allows holders to forfeit their rights.

- *onERC721Received*: a necessary function for code implementing the IERC721Receiver [GTB23a] interface.

The three specific functions are the cornerstones of the Smart Contract and respectively power the three phases:

- *setup*: allows the token owner to initialize the Smart Contract and store their token. It requires the caller to be the token owner and to have approved the Smart Contract as being able to call the ERC721 *safeTransferFrom* function. This function can only be called during the *Initialization* phase.
- *buyToken*: the payable function potential buyers transact with to purchase the token. It is submitted to the `onSale` modifier. It checks for the price, automatically sends the appropriate cuts to royalty holders, transfers its ownership, and puts the token out of sale. This function powers the *Trading* phase.
- *termination*: allows the current owner to retrieve the token, destroying the Smart Contract. This function is only callable if all rights holders have previously forfeited their rights using the retraction function from their respective addresses. Once called, it begins the *Termination* phase.

Before manipulating the Smart Contract, we import various wallets to act as the parties we need (*creator*, *IP holders*, *buyers*, *etc*.). We then mint an ERC721 token, for which we setup an RM-TLSC before having it go through the *Trading* and *Termination* phases. We illustrate the behavior using a test scenario on the Remix IDE. This online IDE allows one to write and test Smart Contracts without requiring lengthy setups. We provide two scenarios in JSON files in the accompanying repository. These files can be imported into Remix's "Run Script" feature and replay a sequence of actions. The first, entitled *scenario-onlySuccesfulTransactions.json* goes through the following series of steps:

1. Constructing the Token Contract,
2. Minting a token,
3. Constructing the capsule,
4. Approving the capsule to handle the token,
5. Initializing the capsule and sending the token,
6. Changing the price,
7. Another address buys the Smart Contract,
8. Changes the price,
9. The first royalty holder retracts,
10. The second royalty holder retracts,
11. Termination.

The primary usage of the Smart Contract is illustrated through an example transaction displayed in Figure 55.

Figure 55: The receipt of an Ethereum transaction to the `changePrice()` function of a Token Level Smart Contract (TLSC).

The second, more complete *scenario.json* goes through the same steps while also trying unauthorized actions (namely, attempted purchases for the wrong price, attempts to change royalty information, and unsuccessful terminations). Figure 56 illustrates the first of these failures.

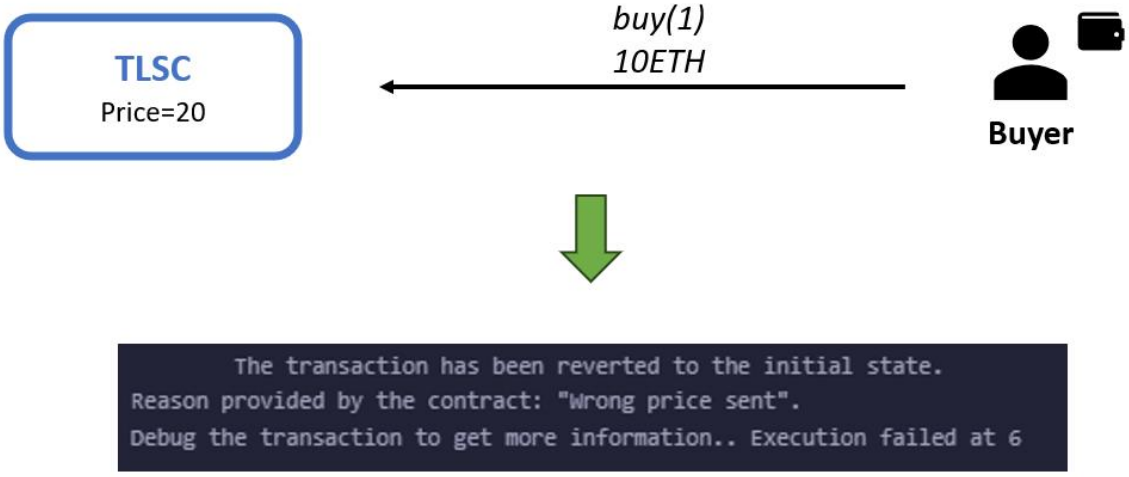

Figure 56: A failed Ethereum Token Level Smart Contract purchase due to the wrong price being sent.

5.III.B. Subsequent use

Now, we make calls and transactions at the RM-TLSC to simulate the use of the solution using the settings shown in Table IX. Here is a test scenario illustrating a basic utilization of the RM-TLSC, including attempted unauthorized actions. It is available to run via SmartPy's "Run Code" button. This scenario illustrates the *Initialization*, *Trading*, and *Termination* phases, and is shown in Figure 57.

```
64    creator = sp.test_account("creator").address
65    IPholder1 = sp.test_account("IPholder1").address
66    fakeIPholder = sp.test_account("fakeIPholder").address
67    randomAddress= sp.test_account("RandomAddress").address
68    buyer1 = sp.test_account("buyer1").address
69    buyer2 = sp.test_account("buyer2").address
70
71    c1 = Royalty(owner=creator)
72    scenario = sp.test_scenario()
73    scenario += c1
74
75    c1.setPrice(3).run(sender=IPholder1, valid=False)
76    c1.setPrice(3).run(sender=creator)
77    c1.addHolder(cut=20, newAddress=IPholder1).run(sender=IPholder1, valid=False)
78    c1.addHolder(cut=20, newAddress=IPholder1).run(sender=creator)
79    c1.addHolder(cut=90, newAddress=fakeIPholder).run(sender=fakeIPholder, valid=False)
80    c1.addHolder(cut=90, newAddress=randomAddress).run(sender=creator, valid=False)
81    c1.addHolder(cut=50, newAddress=randomAddress).run(sender=creator)
82    c1.removeHolder(randomAddress).run(sender=creator)
83
84    c1.transferOwnership().run(sender=creator, amount=sp.tez(3), valid=False)
85    c1.transferOwnership().run(sender=buyer1, amount=sp.tez(2), valid=False)
86    c1.transferOwnership().run(sender=buyer1, amount=sp.tez(3))
87
88    c1.addHolder(cut=20, newAddress=fakeIPholder).run(sender=creator, valid=False)
89    c1.addHolder(cut=20, newAddress=fakeIPholder).run(sender=buyer1, valid=False)
90    c1.setPrice(4).run(sender=buyer1)
91
92    c1.transferOwnership().run(sender=buyer2, amount=sp.tez(3), valid=False)
93    c1.transferOwnership().run(sender=buyer2, amount=sp.tez(4))
94
95    c1.exit().run(sender=buyer2)
96    c1.retraction().run(sender=IPholder1)
97    c1.exit().run(sender=buyer2)
```

Figure 57: A SmartPy test scenario for a Tezos Royalty Managing Token Level Smart Contract implementation.

Lines 64 to 73 constitute the setup of the test scenario and various accounts. Further lines show the *Initialization* of the RM-TLSC, with voluntary failures illustrating the behavior not tolerated by the RM-TLSC. For instance, the transactions on lines 75 and 77 will fail because *IPholder1* has no right over the Smart Contract, whilst the transaction on line 80 will fail because the total shares cannot total more than 100%. The latter is shown in Figure 58.

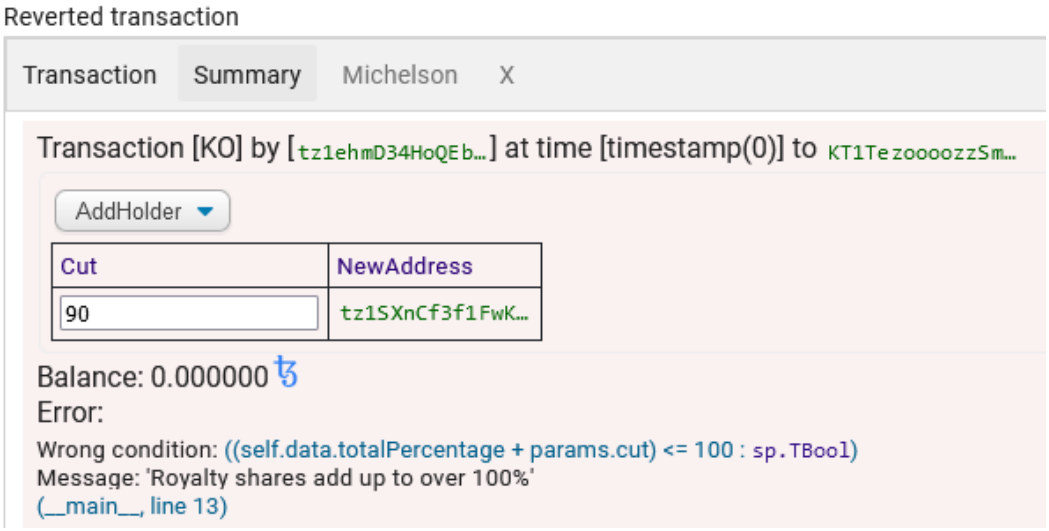

Figure 58: A failed attempt at adding royalty information to a Token Level Smart Contract from a Tezos user.

Lines 84 and 85 show attempted purchases by the current owner and for the wrong price, while line 86 launches the successful purchase of the TLSC by *buyer1*. The details of the transaction, shown in Figure 59, confirm that 20% of the price were indeed sent to *IPholder1*'s address and that the owner's address changes appropriately to *buyer1*'s.

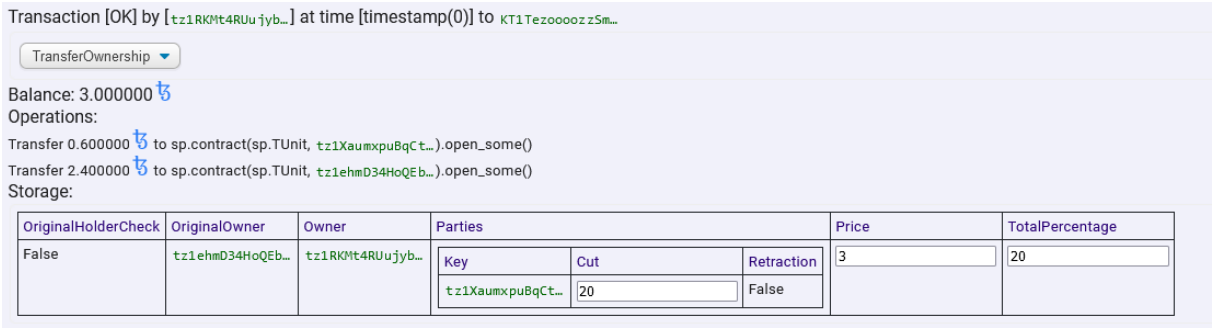

Figure 59: A successful Token Level Smart Contract purchase showing automatic royalty payments on Tezos.

The block between lines 88 and 90 shows that the only field modifiable after the first purchase is the current price. Even the *creator* cannot modify the information once the first purchase passed. Another buying cycle then occurs, landing the token in *buyer2*'s possession while still compensating *IPholder1*. The creator does not get compensated for this transaction due to their absence as a royalty holder. Finally, the last block shows the *Termination* phase. The attempted withdrawal of the token on line 95 fails because the *IPholder1* address has not signaled its retraction. Once it does in line 96, shown in Figure 60, the current owner *buyer2* can terminate the TLSC.

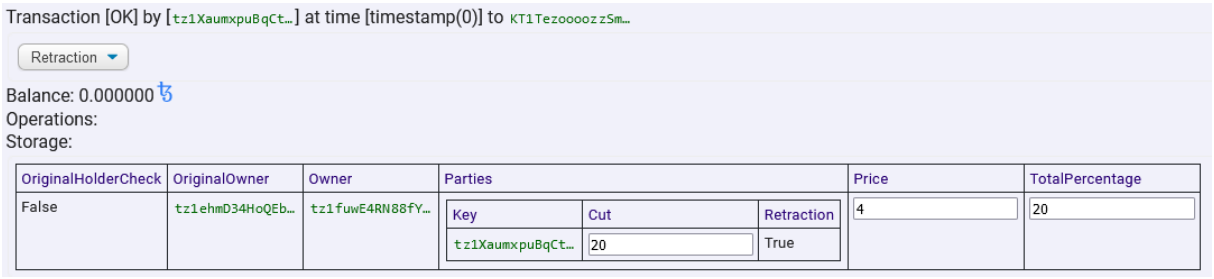

Figure 60: The successful retraction of a royalty holder in a Tezos Token Level Smart Contract.

The Smart Contract used in the Subsection can be imported in SmartPy via: https://legacy.smartpy.io/ide?cid=QmWUHjpqiJsZUW5mfgqCeSgYvxjp6dvdYHALxAjDWXiFet&k=bb8f0c5515cf6cd104b8

5.III.C. Analysis

This section provided implementations for the TLSC, a solution built as a *backward-compatible*, *marketplace-agnostic framework* that ensures *indeterminate* and *systematic* IP limitations and royalty enforcement according to *rules* that are *transparently set* beforehand and allow *cordial retraction*. It also demonstrates the flexibility of the TLSC with respect to blockchains and its independence from marketplaces. Note that the case study has been successfully performed without reference to a specific marketplace, while blockchain specificities are limited to their programming language. The tools used in this paper can be found across most application-driven blockchains, enabling this solution to be quickly implemented across varied environments. The software structure presented here is also modular, allowing users to build upon base functionalities to tailor the solution to their needs. For instance, we took the approach to pay the royalties in real-time, but staggered or on-demand payments are also possible with minor changes.

While providing these beneficial features and being by design open to further extensions, the TLSC still has limitations and shortcomings, some of which are inevitable in the blockchain environment. Many of the issues our approach can suffer from have affected tokens during their uprise. Hereafter, we discuss ten such aspects: two deal with the very nature of digital assets, three deal with the specific usage of tokens, three put our approach in perspective with users and use cases, and the final two come back to the big picture of royalty solutions in blockchain environments.

① The notion of ownership: The TLSC framework relates to the idea of owning a token, given that rather than owning the token directly, users own a Smart Contract that, in turn, owns the token. This has *a priori* consequences on the method acceptance, but, as mentioned in Section III.A., tokens have fueled the debate of ownership by themselves, especially in digital art [REN21]. Furthermore, one must remember that token ownership is reflected in a Token Contract ledger, not by the local storage of software.

② NFT advertisement: This system does nothing to advertise the tokens. This task falls back on the author or a third party. Yet, marketplaces do not use their Swap Contracts to advertise but do it on web platforms. A community or third-party-driven website could step up to the role third parties already have in the space.

③ Royalties over tokens: Regarding whether digital assets should be subject to royalties, we will not go further than the value of assets being contingent on the rules established by the creator, as discussed in Section III.D. Moreover, complex features such as exceptions in applicable transactions, multiple owners, *etc*., can be added without impeding the fundamentals of the solution.

④ Transfer/purchase loophole: In some use cases, this loophole (discussed in Section III.C.) is of little concern; in others, it defeats the very purpose of an automatized framework. The TLSC does not fall victim to it, compromising by systematically enforcing royalties. We then enabled the cordial retraction feature to add flexibility regarding the systematic nature of the TLSC. In particular, the examples in Section V allow owners to set the TLSC's price to zero, thus allowing them to move assets between their wallets. This system could be abused but prevented with a minimum price clause.

⑤ Technical knowledge is required for adoption: The proper initialization of this solution does require a level of familiarity with Smart Contract development, especially in the case of specifically tailored rules. But few tokens are initialized by lone developers sending web3 requests to their Smart Contracts from terminals. The space has evolved to automatize token minting and provide graphically intuitive solutions for the public to create tokens themselves [OPS23], [OPZ23a].

⑥ Integration with existing solutions: From the general point of view of NFT integration, the TLSC has been designed under the requirements of backward compatibility. Yet, from the specific point of view of the first integration step, namely integration to the users' wallets, a difficulty is encountered: wallets blockchain users are accustomed to are not yet suited to reflect indirect ownership. This issue could be solved alongside acceptance, as it did with tokens. Indeed, third parties have had to create wallet solutions around the birth of token standards in their time.

⑦ Third-party exploitation: The Smart Contract does not intrinsically prevent a malicious user from storing a token they purchased and trying to receive undeserved royalty compensation. Moreover, once purchased, the owner has that power, and further buyers can refuse purchases if abusive conditions are in effect. The Smart Contract can also solve this potential issue by only accepting the underlying token's creator as an

author. Tokens themselves have included standards to avoid similar behavior [VOG15], [ENT18], [TEZ23a].

⑧ Other TLSC use cases: The notion of a rule-enforcing, zero-trust third party can be used for many different applications than royalty management. The TLSC paradigm can apply to any use case requiring complete lifecycle rule enforcement. For instance, some media content could be restricted in access regarding addresses or time [ALL21a]. In this case, a TLSC could ensure the content producer can establish rules that will be automatically enforced from the onset. This solution can also be put in perspective with the Metaverse, where the exchange of digital assets is a central paradigm, and automatic royalty payments would contribute to the overall trust of actors. Moreover, web 3.0 use cases in general could enormously benefit from the controlled access of restricted content enabled by a TLSC. A detailed analysis of the foundational notions of the Metaverse can be found in [WAN22], while challenges of the social aspect of the metaverse can be found in [WAN23a]. The TLSC's automatic rule enforcement can also benefit the myriad of applicative verticals such as uncrewed vehicles [WAN23b], smart grid [MOL20], pile sharing [WAN21b], or charitable donations [CHA22a], amongst others.

⑨ Security/zero trust: The TLSC's security is ensured by the blockchain's native security mechanisms, as discussed in Section II.A. We tested abuses at the functional level, attempting forbidden actions throughout our tests (Section V). Given the solution's flexibility, further use and standardization efforts could patch any newly discovered flaws. TLSCs are meant to act as the only required third party, making complex royalty management a one-time setup while featuring zero trust and a high level of decentralization, putting the control in the hands of creators. The history of blockchain teaches us that security faults often lie in third-party software and social engineering attacks. Aware of this fact, we built the TLSC by removing the need for a trusted third party and by requiring the user's explicit approval for the first token transaction. Security risks on the blockchain can also come from within, as assets can be lost or stuck indefinitely in the case of errors. As such, we added the flexibility to cancel and destroy an RM-TLSC in the case of mistakes or upon the realization that this framework is no longer the most appropriate solution. In short, the TLSC paradigm remains safe due to its blockchain nature, granted that users stay in control of their private keys.

⑩ Gas cost: Not only does the initialization of a separate Smart Contract add gas costs, but each transaction launches a series of automatic sub-transactions whose gas needs to be paid for (the examples in Section V could use a cost of zero because of the nature of test environments). This factor can be limited by on-demand or staggered payments, but the initialization of the Smart Contract cannot avoid adding cost. While off-chain structures must rely on paying royalty managing companies or specialized departments, on-chain actors must accept that extra functionalities come at an expense: gas.

As made evident by the above observations, future work regarding this solution lies in its integration with commonly used daily blockchain solutions (*e.g.,* wallets, marketplaces) and in the ease of instantiation, which requires familiarity with blockchain development. A possible approach to solving the latter issue could take inspiration from the automatic Smart Contract generation solution brought forth in Section II.

5.IV. Smart Contract load balancing

In this Section, we walk through and discuss illustrative implementations of the workflow and architecture brought forth in Chapter 4, Section II.B. Access control not being a central feature of this paper, we set out two 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. As in the previous sections, we shall begin by illustrating the Smart Contract's design on the Tezos infrastructure before moving on to a complete implementation on Ethereum.

5.IV.A. Smart Contract design

This Subsection will focus solely on the Smart Contract, using a simple, nondescript, illustrative scenario. As a reminder, this Smart Contract powers the `greenLight()` database verification function by recording a map of fingerprint hashes and appends new entries to said map on demand. Figure 61 and Figure 62 show the SmartPy functions, which are lightweight and intuitive, as required by blockchain development.

```
def __init__(self, owner):
    self.init(
        fingerprintHashes = sp.map(tkey = sp.TString), owner=owner
    )

@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]
```

Figure 61: The SmartPy Smart Contract entry point functions which manage the on-chain database of our load balancing mechanism.

The two functions shown in Figure 61 (excluding the constructor `__init__`) will manipulate the map of hashes sent by the app in a complete use case. The first adds its hash parameter alongside some information after verifying the request is sent by the creator and that the hash is not recorded previously. This last check can be removed if overriding were allowed. The second function simply deletes an entry if requested by the owner. These are the only two functions that write information on the blockchain (hence the *@entry_point*).

```
@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 62: The SmartPy Smart Contract view functions which pass on information the App during the load balancing process.

The three functions shown in Figure 62 are the *get* functions which pass on information to the App. They only read data and are hence preceded by *@on-chain_view*. They send back the existence of a given hash in the map, the size of the map, and the info associated with a given hash, respectively. Now, we build a test scenario that will use this Smart Contract. It is shown in Figure 63, and line numbers will refer to it for the remainder of IV.A.

```
38  scenario = sp.test_scenario()
39  creator = sp.test_account("creator").address
40  otherUser = sp.test_account("otherUser").address
41  c1 = FingerprintStorage(owner=creator)
42  scenario += c1
43
44  c1.addToDB(hash="0x0001", info="first input").run(sender=creator)
45  c1.addToDB(hash="0x0001", info="same input").run(sender=creator, valid=False)
46  c1.addToDB(hash="0x0002", info="unauthorized user").run(sender=otherUser, valid=False)
47  c1.addToDB(hash="0x0002", info="second input").run(sender=creator)
48  c1.addToDB(hash="0x0003", info="third input").run(sender=creator)
49  scenario.show(c1.getInfo("0x0003"))
50
51  c1.addToDB(hash="0x9999", info="input including a mistake").run(sender=creator)
52  c1.deleteEntry("0x9999").run(sender=otherUser, valid=False)
53  c1.deleteEntry("0x9999").run(sender=creator)
54
55  scenario.verify(c1.getSize() == 3)
56  scenario.show(c1.compare("0x0001"))
57  scenario.show(c1.compare("0x0002"))
58  scenario.show(c1.compare("0x0003"))
```

Figure 63: A SmartPy load balancing Smart Contract test scenario initialization with two users, having full rights and no rights, respectively.

Lines 38 to 42 show the initialization of the test scenario, include the addresses to `creator` and `otherUser`. The second set of transactions, lines 44 to 49, attempt to initialize the on-chain database as the App would do. All but the second and third transactions succeed, while these two are correctly identified are erroneous inputs. The first would be a duplicate input, while the second it requested by an unauthorized user.

These transactions are subsequently denied by the blockchain (or *reverted*), which is illustrated in Figure 64.

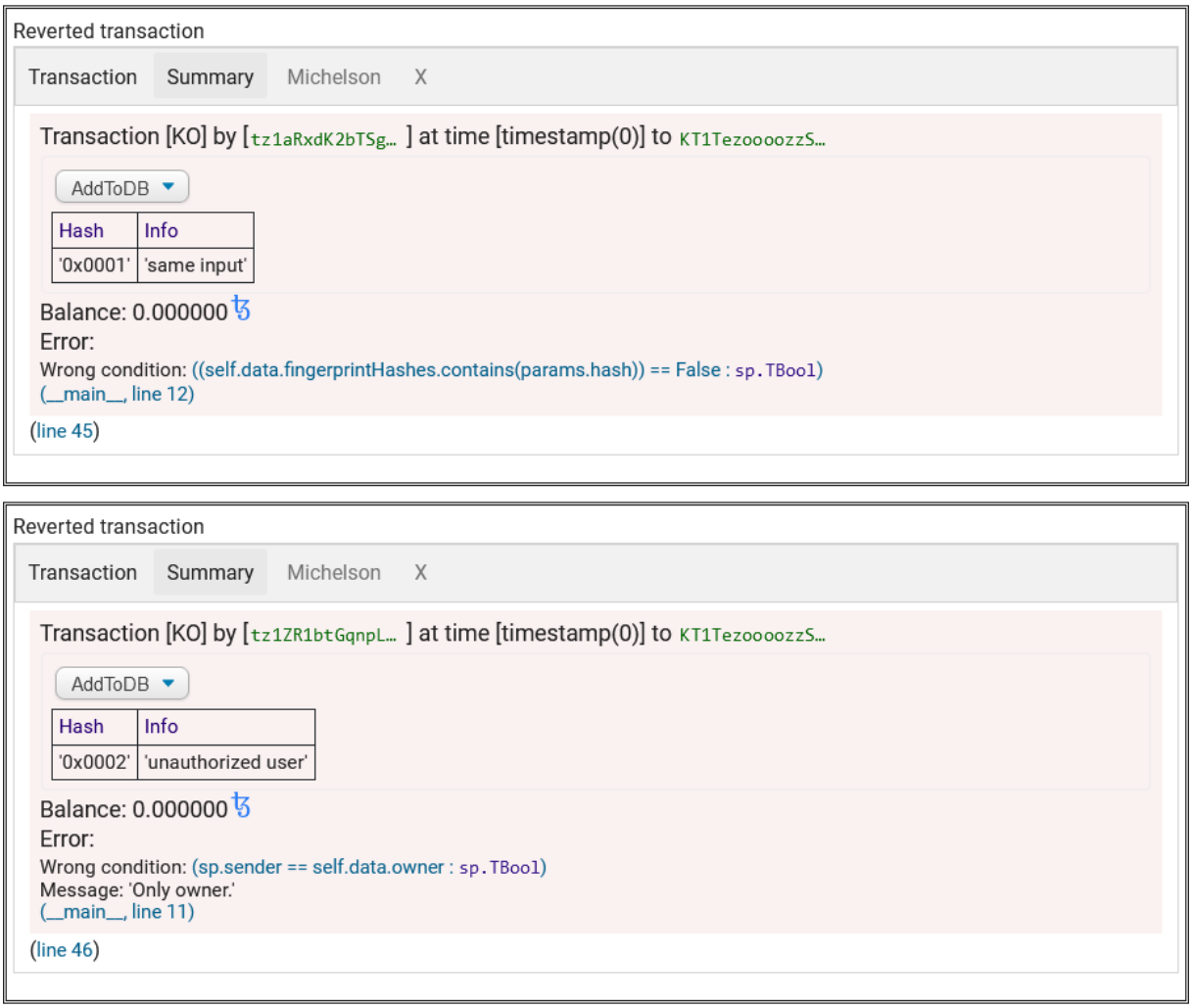

Figure 64: The SmartPy test scenario designed initialization failures, seen from the Smart Contract's perspective.

When `creator` requests the addition of a new, properly formatted input the transactions are executed without issues. At the term of line 48, the Smart Contract's storage matches what is shown in Figure 65.

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

Figure 65: The SmartPy Smart Contract storage after the addition of the three inputs in lines 44, 47, and 48 of Figure 63.

Further, the addition of new inputs can be prone to errors, which we illustrate in line 51, where an unwanted entry is added by `creator`. The next transaction shows `otherUser` cannot correct this mistake, and the following shows `creator` can. The Smart Contract's storage corresponds to Figure 66 before the deletion, and back to Figure 65 after.

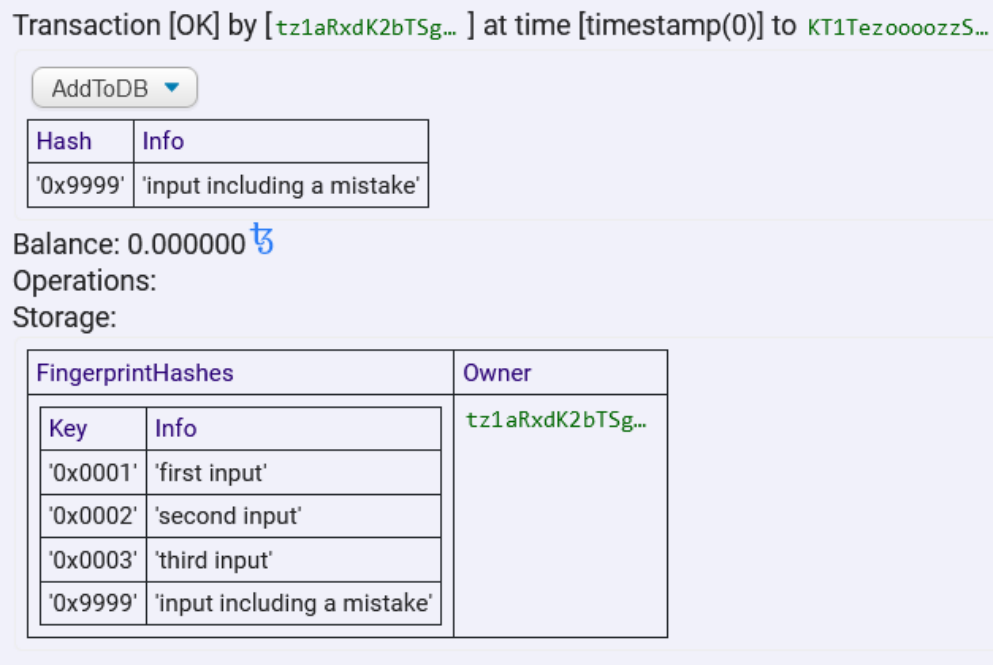

Figure 66: The SmartPy test scenario showing an unwanted entry being added (top) and subsequently removed (bottom).

We consider storage of entries 0x001, 0x002, and 0x003 to correspond to the initial storage of a given use case. When the App executes the `greenLight()` function, the operations from lines 55 to 58. The App would start by checking that the number of recorded hashes is equal to the number of entries it has in its local storage (3 in this case) and then use the compare function to check the hashes of our recorded fingerprints appearing on the blockchain. If this test passes, the `greenLight()` function returns *True,* thus ensuring that the off-chain and on-chain databases match.

The Smart Contract used in the Subsection is available on SmartPy via [SPY23c].

5.IV.B. App design

In addition to the Smart Contract detailed above, the architecture requires an off-chain database and an App. In the context of an academic study, we opted for a local storage database, which is possible because of the tamper-proof qualities brought by the solution. Regarding the App, we decided to use Python alongside the web3py library to interact with our blockchain and Smart Contract. Note that given the specifications of our private

blockchain, we had to inject a middleware solution, as explained in [ETH23b]. Any programming language supporting a library that enables blockchain communication could have been used to a similar avail. The complete App we used is available at [GTB23b] and was only modified to accommodate various fingerprints tailored to the different use cases. It also contains the code shown in the following screenshots.

The App begins by instantiating the account it will operate and establishing a proper connection with the blockchain and the deployed Smart Contract before calculating the input's fingerprint. The fingerprinting method is called from a separate file so as not to have to modify this primary function when changing methods and use cases. The exact actions to take largely depend on the library in context. A web3py-appropriate example is shown in Figure 67 and forwards.

```
def mainCall(scene, defaultAccount=default_acc, threshold=0.7):

    newFP = GetFingerprint(scene)
    #------------------------------------------------------------------------------ Getting SC
    w3 = Web3(Web3.HTTPProvider('https://services.blockchainsecure.io/dev/alex1/ethsigner1'))
    #w3 = Web3(Web3.HTTPProvider("https://rinkeby.infura.io/v3/980ac657631e4cecb0ad88385d21852a"))

    w3.middleware_onion.inject(geth_poa_middleware, layer=0)


    private_key1 = os.getenv('PRIVATE_KEY')
    account1 = w3.eth.account.privateKeyToAccount(private_key1)
    w3.eth.default_account=account1.address

    ####

    if w3.isConnected:
        #w3.eth.default_account=w3.eth.accounts[0]
        w3.eth.default_account=account1.address
        abiFile = open("build/contracts/Fingerprint.json", "rb")
        abiFile=json.load(abiFile)
        FPContract = w3.eth.contract(
            address=getAddress(),
            abi=abiFile["abi"]
        )
    else:
        print("web3.isConnected failed")
```

connects to the blockchain

loads the Smart Contract

Figure 67: The main function of the Python App in our load balancing architecture.

The App then executes the `greenLight()` by verifying that the sizes of the databases match and that local fingerprint hashes can be found on-chain, as shown in Figure 68. Notice that these operations are not transactions but calls (much faster and do not cost gas).

Figure 68: The `greenlight()` process from the App's perspective.

Then comes the primary processing: under the condition of a `greenLight()` validation, the input's fingerprint is compared to local entries, which determines the operation path: copy, near-copy, or no-copy. The details of the steps are explained in Chapter 3, and their implementation is shown in Figure 69.

Figure 69: The App's processing according to the process diagram shown in Figure 42.

Only a no-copy case, where the input was not found to be correlated with any entry, calls the add function. The `add()` function establishes the required connections and builds a properly formatted transaction to store the new hash in the Smart Contract. The essential part of this operation is summarized in Figure 70.

```
#adds hash to Smart Contract
transaction1 = FPContract.functions.addToDB(hash, info).buildTransaction({ 'gasPrice': 123456,'gas' : 2000000,
                                                                            'nonce' : w3.eth.get_transaction_count(account1.address)})
signed_tx1 = w3.eth.account.sign_transaction(transaction1, private_key1)
txn_hash1 = w3.eth.send_raw_transaction(signed_tx1.rawTransaction)
txn_receipt1 = w3.eth.wait_for_transaction_receipt(txn_hash1)
print(txn_receipt1)
print("Added hash to onchain database. Transaction number: "+ str(txn_hash1.hex()))
```

Figure 70: The building of a database addition transaction to the Smart Contract.

5.IV.C. Subsequent use

We shall now put our architecture before a filter-based near-copy detection of mirflickr25k [MIR23] images. To test the implementation, we will compare the entries stored in the off-chain database to artificially created (more or less) near copies. We created modified versions of the original inputs by subjecting them to standard image processing attacks, namely:

- Conversion to black and white,
- Brightness increases,
- Cropping (50%),
- JPEG compression at a quality factor of $Q = 90$,
- Resizing to 600x400,
- And combinations of these alterations.

Figure 71 shows images before and after such modifications. In this example, they were subjected to cropping, brightness increase, and resizing, respectively.

Figure 71: The before (top) and after (bottom) of images been subjected to image processing attacks (resizing, luminosity increase, format encoding modification).

These modified versions are given to the architecture as inputs. Naturally, some will be semantically so close to one of the entries that the App will categorically refuse them. In contrast, others bear so little resemblance to originals that they will be considered original inputs. The border and scale between these answers lie in the selection of the fingerprinting method and threshold.

We selected the International Standard Content Codes (ISCC) [ISC23a]. The ISCC is a complete ISO/AWI 24138 standardization work item whose goal is to provide an open-source, cross-sector, universal identifier of different kinds of content. It is also a lightweight and similarity-preserving fingerprinting method for our purposes. An interesting feature of the ISCC is its capacity to generate similarly formatted outputs from completely different input formats. The media identifier is universal and enables databases of images, text, video, audio, *etc*., to be treated uniformly. The ISCC was designed for blockchain-based registration; it is short (between 13 and 55 characters), so we forego the hashing of the code and store the code directly in the Smart Contract. ISCC codes are composed of 4 parts: their Metadata code, Content code, Data code, and Instance code, which can be used standalone or in combination.

This section only worries itself with Content codes, while other use cases could take full advantage of its different facets. Our architecture can compare total ISCC codes and make decisions based on separate processing. For instance, we could require a strict threshold of differences between Content codes while enabling a looser check for Metadata and Instance codes to differentiate original content from other users or with different encodings. We decided to set the near-copy detection threshold as a Hamming distance of 10 for general detection purposes, *i.e.,* fingerprints with a Hamming distance of 10 or less will be considered near-copies.

The partial content flagging feature of the ISCC could also be put forth to identify copied content being used within other content. This feature is illustrated in Figure 72 [ISC23b]. This feature makes near copy detection even more critical as content inserted in other content is naturally modified. This notably opens up to use cases dealing with fake news.

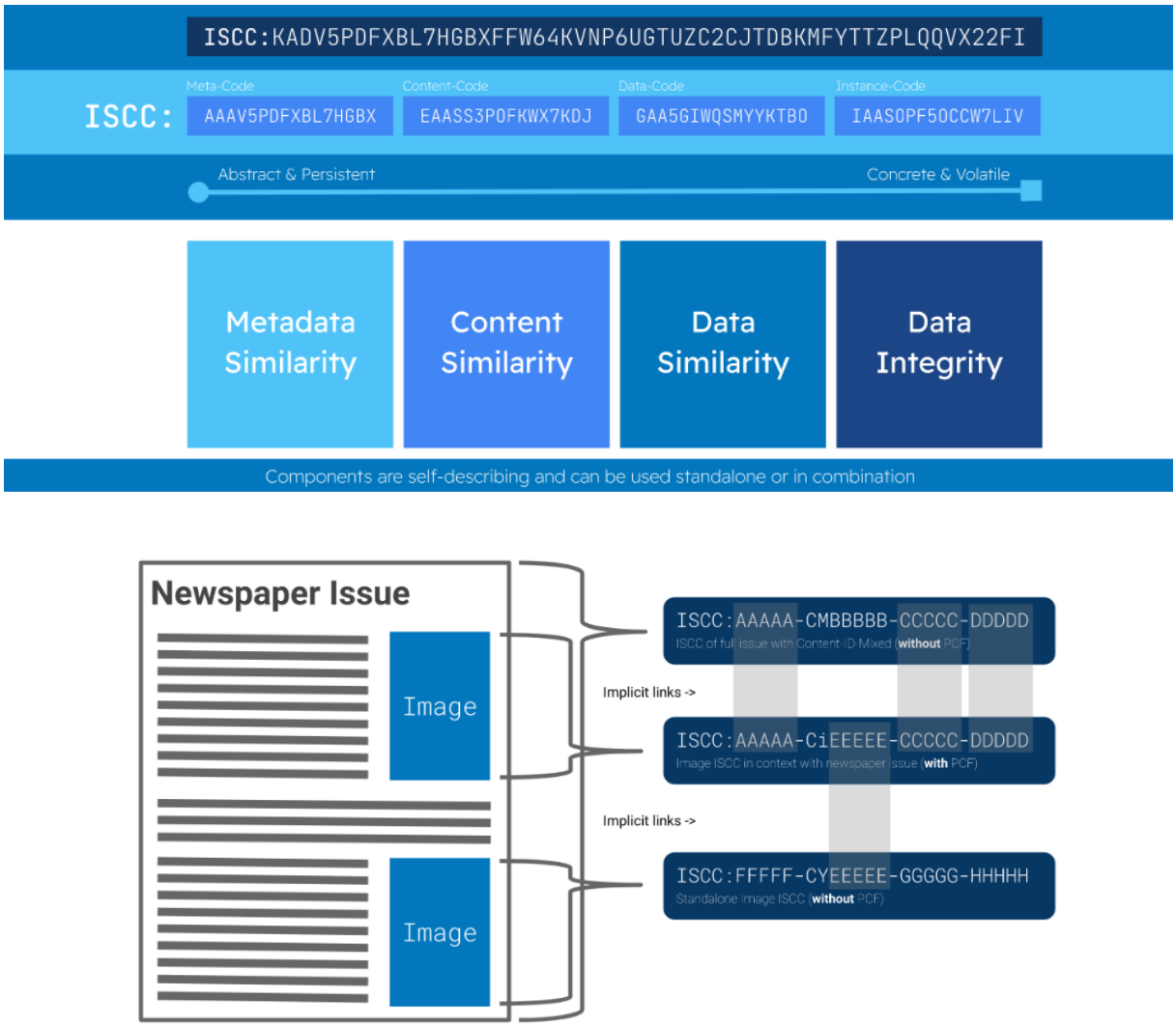

Figure 72: Basic structure of the International Standard Content Code [ISC23b] (left) and illustration of its content flagging.

We find ourselves in the first scenario, where the initial images are fingerprinted in the database, and the Smart Contract has previously been deployed. We will now follow the workflow with each of the three possible cases: copy, near-copy, and no-copy.

The copy case is the most direct. The input is fingerprinted, and said fingerprint perfectly correlates to a database entry (which is blockchain-backed). The fact that the correlation is perfect is of little regard, as the blockchain is called to ensure the hash of the proposed file is found as is in the Smart Contract storage. The input is then refused. This case is illustrated in Figure 73, with an input simply being a pre-recorded file.

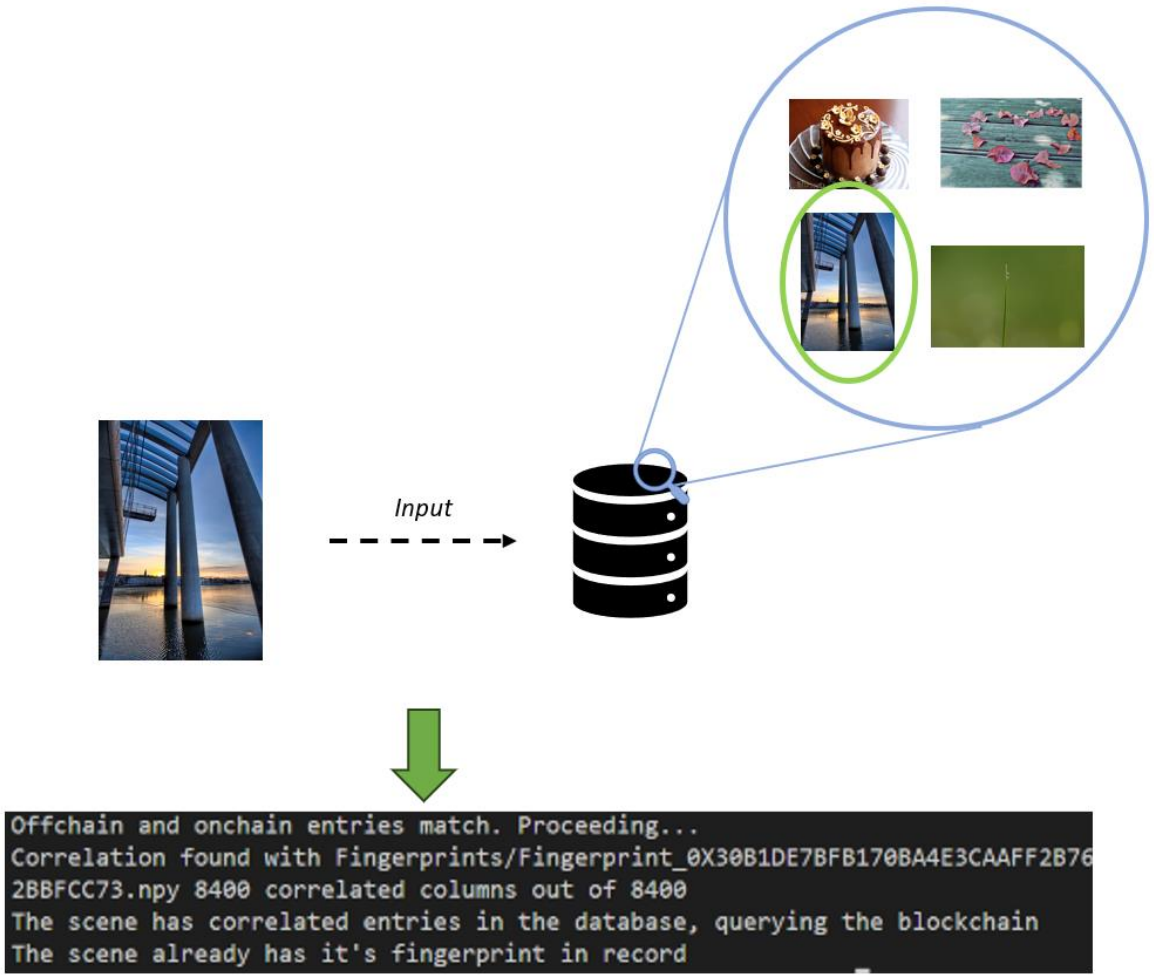

Figure 73: The output of the process when given an already recorded entry as input.

The near-copy case is very similar. The input is fingerprinted, and this fingerprint does not pass the threshold of originality. The hash of the fingerprint not appearing as is in the Smart Contract's storage does not matter: the input is refused. This case is illustrated in Figure 74. We used an altered version of a previous input, also shown below.

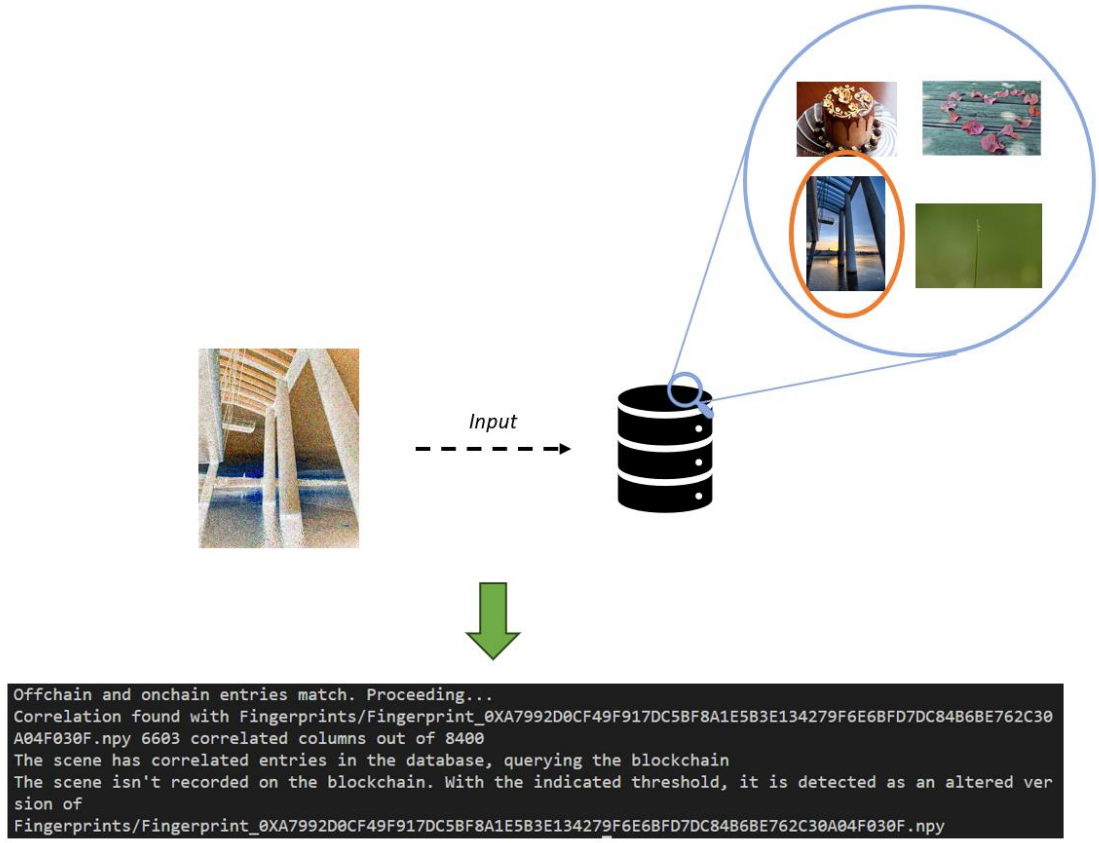

Figure 74: The output of the process when given a modified version of a recorded entry as input.

Now, for the no-copy case, we offer a new random image for the mirflikr dataset as input to the App. This input's fingerprint is not correlated with any of the recorded fingerprints and is hence considered an original input. The App offers a prompt to add this input as a database entry (this process could be done automatically, but we decided to prompt database additions), which can be denied with no follow-up or accepted. If accepted, the Smart Contract's `addToDB()` function is transacted with, using the new fingerprint's hash as input data. The confirmation of this transaction makes the information immutable, and hence, the off-chain database is updated with this latest entry. This case is illustrated in Figure 75 and may be cross-checked using a Rinkeby explorer such as [RIN23].

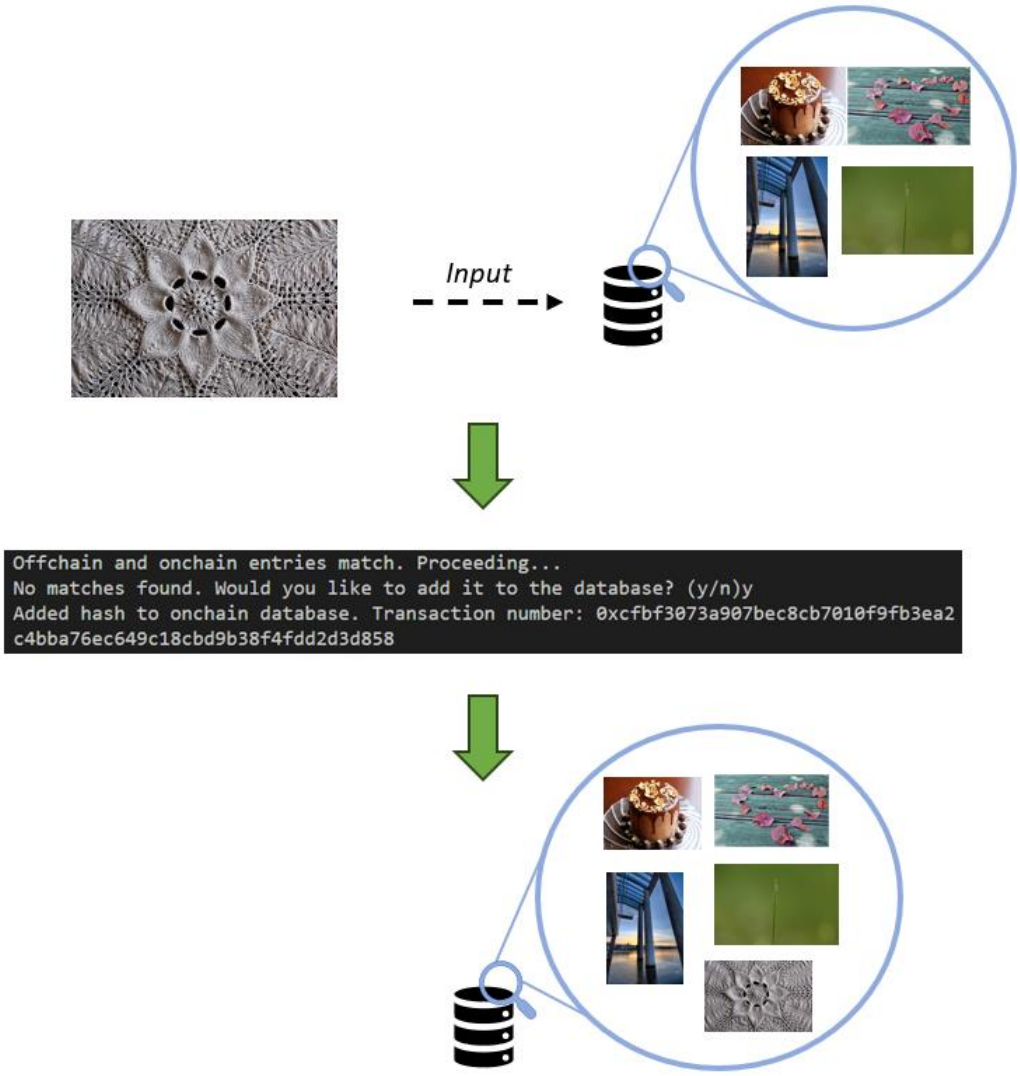

Figure 75: The successful addition of a new entry (detected as original) in the blockchain-backed database.

All the above operations include a preventive `greenLight()` from the Smart Contract. If a malicious user were to gain access to the database and delete an entry from the records for their entry to be perceived as semantically original, the `greenLight()` function would not permit the App to function. We acted as such a user, and the result is shown in Figure 76. The same modification cannot happen with the verification database, as it appears on-chain and is subsequently unalterable.

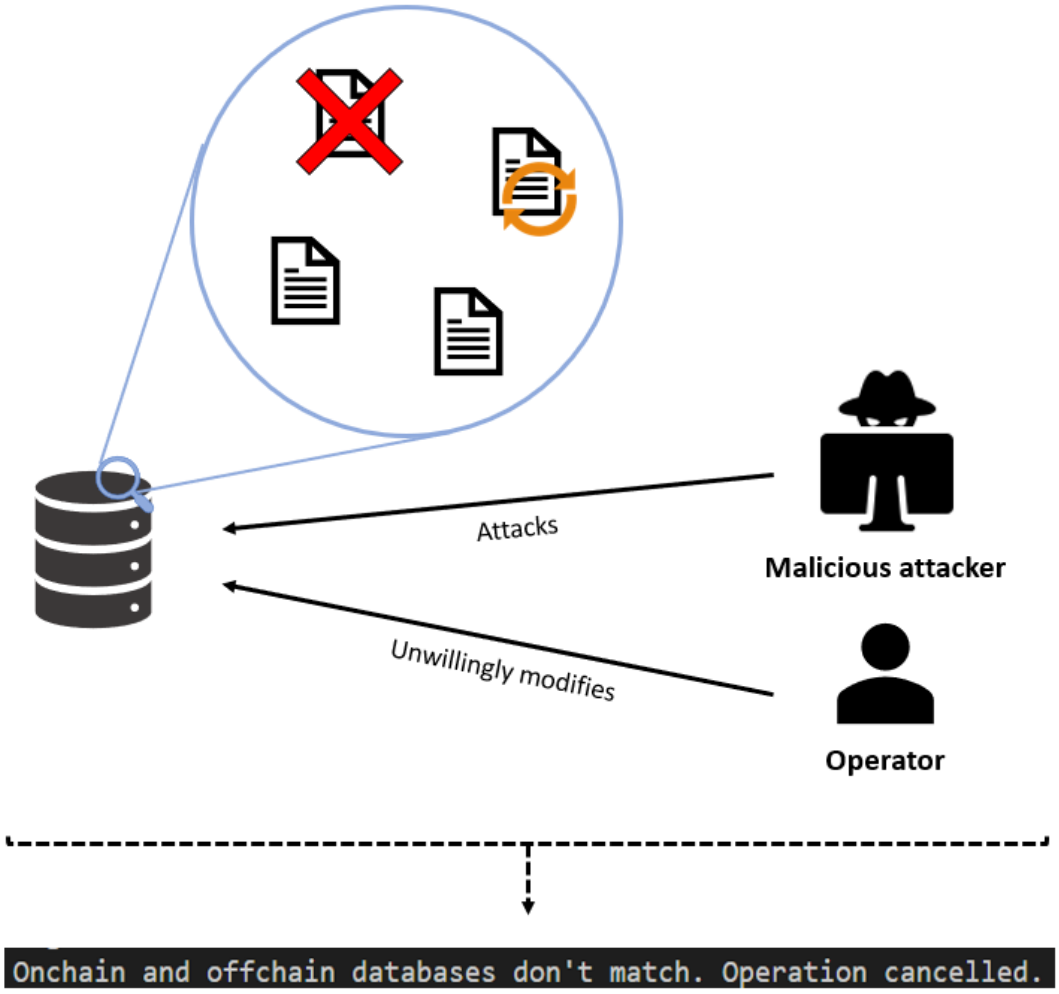

Figure 76: The result of the process being ran after the database has been tampered with i.e., a `greenlight()` function failure.

5.IV.D. Analysis

This section covered the details in our architecture specification and a step-by-step workflow in action for a broad use case. As supported by the methodological discussion, the architecture is very flexible in accommodating various technologies. This includes blockchains, database solutions, App languages, fingerprinting methods, *etc*. We inspect the advantages provided by our approach through the lens of workflow genericity, resource usage, and interoperability:

① **Workflow genericity:** The workflow advanced in the paper provides a robust structure catering to the needs of verifiable data integrity. It does so by making the best of the mutually beneficial association of on-chain and off-chain technologies. It also enables the variable processing of multiple forms of multimedia content. As demonstrated in

Section 4, the fingerprinting method used in the workflow can easily be modified without affecting the other parts of the algorithm. As such, one could use this methodology to track photography being copied with filter modifications or monitor metadata changes in audio files.

As stated in the introduction, this paper analyses the governance mechanism of the proposed architecture. As such, we will not provide detailed reports on near-copy detection performance (as the fingerprinting methods themselves purely determine it) nor on security mechanisms. Briefly:

- The architecture does not impose supplementary constraints to multimedia processing, and as such, the analysis provided by the developers of the specific methods stays relevant.
- We take advantage of the native security features brought by the blockchain environment. As such, the advanced workflow can only be as secure as the link between the App and the blockchain.

② **Resource usage:** This section investigates the overall computational load required by our workflow.

Although the App requires computation, it is naturally substantially inferior to the computation of the Smart Contract. The former is also largely dependent on the fingerprinting method, unaffected by the rest of the processing. As such, the performance analysis of fingerprinting methods brought forth by their authors remains relevant and was corroborated by our tests. Hence, we focus on the blockchain computational load. Computation on the blockchain can be measured with gas. The gas fee is defined by the cost (in local cryptocurrency) that users pay blockchain validators. Each transaction costs a certain amount of gas, and a given block has gas limits that restrict what can be executed within a block. We also must remember that execution times and gas costs will vary significantly depending on the blockchain in context and the network traffic at the time of execution, as these costs are dynamic. As such, the results we provide here are linked to their context and execution conditions.

First, Smart Contracts are simple and, hence, lightweight. The total storage of the Smart Contract is decided upon deployment and can be estimated via diverse tools such as Tezos' client-server protocol (RPC). This helps generate values for fees, gas limits, and storage values. These automatically generated amounts, as well as the deployment figures that can be found on an explorer, are shown in Figure 77. They confirm that the design philosophy of the workflow is respected by remaining small. Of course, practical applications need to account for data they will store as hashes in the Smart Contract and add that value to the storage limit of their code. At the time of writing, *tez* (the native Tezos token) is worth around one USD, making this deployment cost less than 22 cents. Implementations accounting for storage would cost more, but stay very affordable, especially on blockchains with low gas costs such as Tezos.

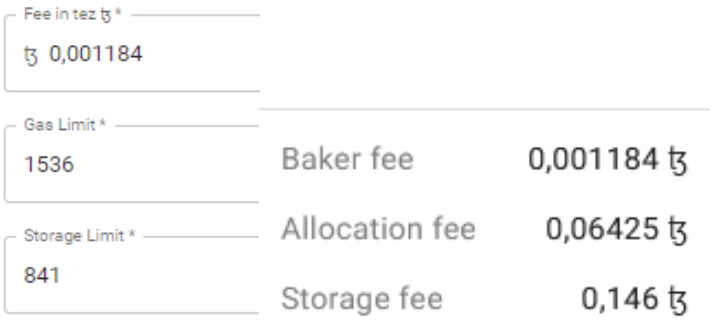

Figure 77: Tezos Smart Contract Remote Procedure Call estimates for deployment (left) and deployment fees (tight).

Regarding our Ethereum implementations, the details of the initial deployment of the on-chain programs can be found in Figure 78 (the deployment of the Smart Contract is the same for both use cases). It shows single block deployments of the Smart and Token Contract, respectively, using 15.24% and 61.45% of gas limits (set by default at 4.5 million), for a total of 0.03451321ETH (for a gas price of 10 Gwei, or 10^{-8}ETH). Use cases not needing the tokenization of their assets can eliminate the latter and only use a single lightweight Smart Contract.

```
Deploying 'Fingerprint'
-----------------------
> transaction hash:    0x68bf8a64d508316b7a415f6057614fbbb4713a881c759c3af9c6319dcec811f2
> Blocks: 1            Seconds: 12
> contract address:    0xA75f207314C85F4891657a2D4f73b19b88b21dc9
> block number:        11390403
> block timestamp:     1663331580
> account:             0x8DCe35025bf87143524A7dC8C7ac1EDA326F281C
> balance:             18.364189687758944911
> gas used:            685852 (0xa771c)
> gas price:           10 gwei
> value sent:          0 ETH
> total cost:          0.00685852 ETH
```

```
Deploying 'NFT721'
------------------
> transaction hash:    0x7c124b23b9a50875dff86a89f111aba9c82ef359abefe392ceaaecf44bb2cd26
> Blocks: 1            Seconds: 12
> contract address:    0xAAFFfFF06a971b57ca87953010135d771B91f965
> block number:        11390405
> block timestamp:     1663331610
> account:             0x8DCe35025bf87143524A7dC8C7ac1EDA326F281C
> balance:             18.336246867758944911
> gas used:            2765469 (0x2a329d)
> gas price:           10 gwei
> value sent:          0 ETH
> total cost:          0.02765469 ETH
```

Figure 78: Deployment figures of the Ethereum Smart Contract (top, previous page) and Token Contract (bottom) on our private blockchain.

For the museum IPR use case, populating the Smart Contract with six entries cost us 0.000114ETH per entry, while the tokenization cost 0.000226 ETH per entry (for a gas

price of 1.5 Gwei). Although this step is the most significant resource sink in the entire process, it stays in the scope of a blockchain application. The gas and time spent scales linearly with the number of entries, so even databases of a few hundred to a few thousand entries could comfortably be processed in a couple of hours. This, of course, depends on the block rate of the blockchain in context.

After the setup and for general use, the Smart Contract is only invoked at two specific moments. This leaves most of the processing to the faster and more efficient app. The first use is the `greenLight()` function. This instance 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. In our experience and with our testing setup, this step only added up to 2 seconds of execution to the processing of an input. The second use is if a new entry is to be added to the database. This step is the initial setup brought to the scale of a single entry. The transaction we executed to illustrate Subsection 4 cost the same amount of 0.000114ETH. As was our aim, this localized and minimal use of blockchain enables us to avoid long processing times and excessive gas fees.

③ **Interoperability:** The notion of interoperability appears in diverse aspects of this workflow. Not only is the infrastructure interchangeable, but the applicative components are also. This enables the solution to be used with state-of-the-art components from different sectors.

In terms of hardware, the off-chain database can be as simple as a cloud computing data storage, implementing no features or modern customization, access management, and data integrity. The same applies to the blockchain. Only minimal tools are used, and these Smart Contracts can be transposed to any modern application-oriented blockchain. Although the language and specific performances in gas and time will differ, they stay relatively uniform. We notice that the code is formatted very similarly in Solidity (Ethereum) and SmartPy (Tezos), that processing times are in the order of seconds for both (affected mainly by block rates), and that gas costs remain low given the simplicity of operations required.

Regarding the software, the only non-trivial requirement for the programming language is a library connecting to the desired blockchain. For instance, web3py [WPY23] allows one to connect to the Ethereum blockchain via Python, web3.js [WJS23] is its JavaScript counterpart, and Taquito [TAQ23] is a TypeScript Library that enables Tezos interactions. The requirements of the fingerprinting technology are even more lenient, as its input can be formatted to the desired length via a cryptographic hashing function. The output information can also be formatted to suit any token standard, making the Token Contract a flexible methodological brick.

Future work regarding this framework could adapt the approach to other web2 multimedia tools and standards. Further, the generality of enabling heavy computation is not restricted to media-related use cases. Contrarily to data, consensus, or network-layer solutions, this purely applicative approach enables a very flexible implementation.

5.V. Asset lifecycle management for visual content

In this section, we combine various solutions discussed in Chapter 4 and implemented earlier in this chapter to provide the full lifecycle support of visual content representing assets in blockchain environments. This is enabled by the interoperability brought at various levels by our methodologies, inherent to the logic they were built around, and by the complementarity of their objectives. We put this combined solution at the service of two real-world use cases. The first will deal with the simulation of a museum being wary of the IP of the content they make available online and will use our load balancing, token prototyping (a simplified version of automatic Smart Contract generation), and TLSC solution while the second will further the MThing live content distribution use case by providing a ZKP aspect to it, and will use our load balancing, automatic Smart Contract generation, TLSC, and Token brokerage solution.

These solutions were deployed and tested on the Ethereum infrastructure detailed in Section I.

5.V.A. Complete web2 content management

In this use case, we implement the architecture presented in Chapter 4's Figure 47 without integrating other high-level operations. 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 [LOU23]. We used 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.

We elected to use a robust video fingerprinting method brought forth in [GAR16]. This method is optimized for live recordings and invariant regarding scale and affine transformations, which are common in copied content. We compared these fingerprints using a normalized correlation method with a threshold of .7, decided upon based on the method's performance [GAR16].

At a macro level:

1. Our load balancing architecture will analyze and authenticate entries as semantically original.
2. Accepted entries will be tokenized to represent underlying IP (*ala* MPEG) in a standardized format using token prototyping.
3. Subsequent assets will be protected using a TLSC.
4. These assets can then optionally be sent to a broker for distribution.

Testing was done similarly to our pure load balancing implementation by creating altered versions of the rooms using the same multimedia attacks as in Section IV. Some before/after examples of frames are shown in Figure 79.

Figure 79: The before (top) and after (bottom) of images belonging to sequences been subjected to image processing attacks (resizing, luminosity increase, format encoding modification).

The initial setup varies from the previous section by the deployment of a Token Contract. We selected the popular ERC721 standard, the metadata of which we will populate with the hashed fingerprint of the file. Many different Token Contracts could be deployed across various blockchains but do not contribute to this use case. The App holds the address of this contract alongside the one for the Smart Contract. The first processing steps follow the load balancing model up to the acceptance of a new entry in the database. Alongside the transaction that appends the latest entry in the Smart Contract, the Token Contract is queried within the same function. The underlying operations and their result are shown in Figure 80.

```
#minting of the token
if w3.isConnected:
    abiFile = open("build/contracts/NFT721.json", "rb")
    abiFile=json.load(abiFile)
    ERC721 = w3.eth.contract(
        address=getERCAddress(),
        abi=abiFile["abi"]
    )
else:
    print("web3.isConnected failed")

transaction2 = ERC721.functions.mintNFT(userAddress, path).buildTransaction({
    'gasPrice': 123456,'gas' : 2000000,  'nonce' : w3.eth.get_transaction_count(account1.address)})
signed_tx2 = w3.eth.account.sign_transaction(transaction2, private_key1)
txn_hash2 = w3.eth.send_raw_transaction(signed_tx2.rawTransaction)
txn_receipt2 = w3.eth.wait_for_transaction_receipt(txn_hash2)
#print(txn_receipt2)
print("Minted token and sent it to "+userAddress)
print("Transaction hash: "+str(txn_hash2.hex()))
```

```
Offchain and onchain entries match. Proceeding...
No matches found. Would you like to add it to the database? (y/n)y
Added hash to onchain database. Transaction number: 0xcfbf3073a907bec8cb7010f9fb3ea2
c4bba76ec649c18cbd9b38f4fdd2d3d858
Minted token and sent it to 0x8DCe35025bf87143524A7dC8C7ac1EDA326F281C
Transaction hash: 0x38d8e987e5bcd8459e6d9e31fef4b5b0f7bd7e1e44d346ac794870ca67f980ca
```

Figure 80: App's connection to the Token Contract and building of a minting transaction (top) and result of the the succesful addition of an entry into the authenticated database and subsequent minting of the token representing the process (bottom). These operations were ran on our private blockchain.

Within this use case, the token is sent to the user directly, as it is meant to represent their IP and bears the trace of the previous authentication process. They are then free to keep it as proof or exchange/sell it as they see fit. Before any potential sales, the ERC721 token is set into a TLSC that is initialized with the ruleset defined by the museum, which can restrict the token's price and exchange frequency and receive royalties for all subsequent sales. Please note that the burning (or deletion) of the token created alongside database inclusion does not occur within this proof-of-concept.

This use case represents the direct application of some of the methodologies we brought forth and shows they can be combined effectively. We shall add an extra layer of complexity with a use case requiring the integration of this combination with other high-level solutions deployed for complex use cases.

5.V.B. Complete MThing content management

We now illustrate how our architectures and workflows can upgrade and automate existing high-abstraction solutions via the example of MThing content distribution we explored throughout Chapter 4 and Section II earlier in this chapter. The goal is to have more control over the entire process and lifecycle of the data put up for sale. Although functional, the original workflow can benefit from the workflow presented in this paper. While this paper's innovation could support the MThing Smart Contracts in the control of FTs (*e.g.*, by recording previous buyers and preventing double access), the main advantage it can provide resides in the minting and authentication of an NFT standing for the final admission of the content, which can further be used as a Zero-Knowledge Proof (ZKP) [GOL94]. In [ALL21a], 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 limit access to targeted content. By implementing our workflow, the access of this content can be proved and traced in an authenticated fashion, not only through the specification of access tokens that can be limited to given addresses and timeframes but also in the creation of an NFT timestamping the access to the content. The various solutions can be connected, as shown in Figure 81.

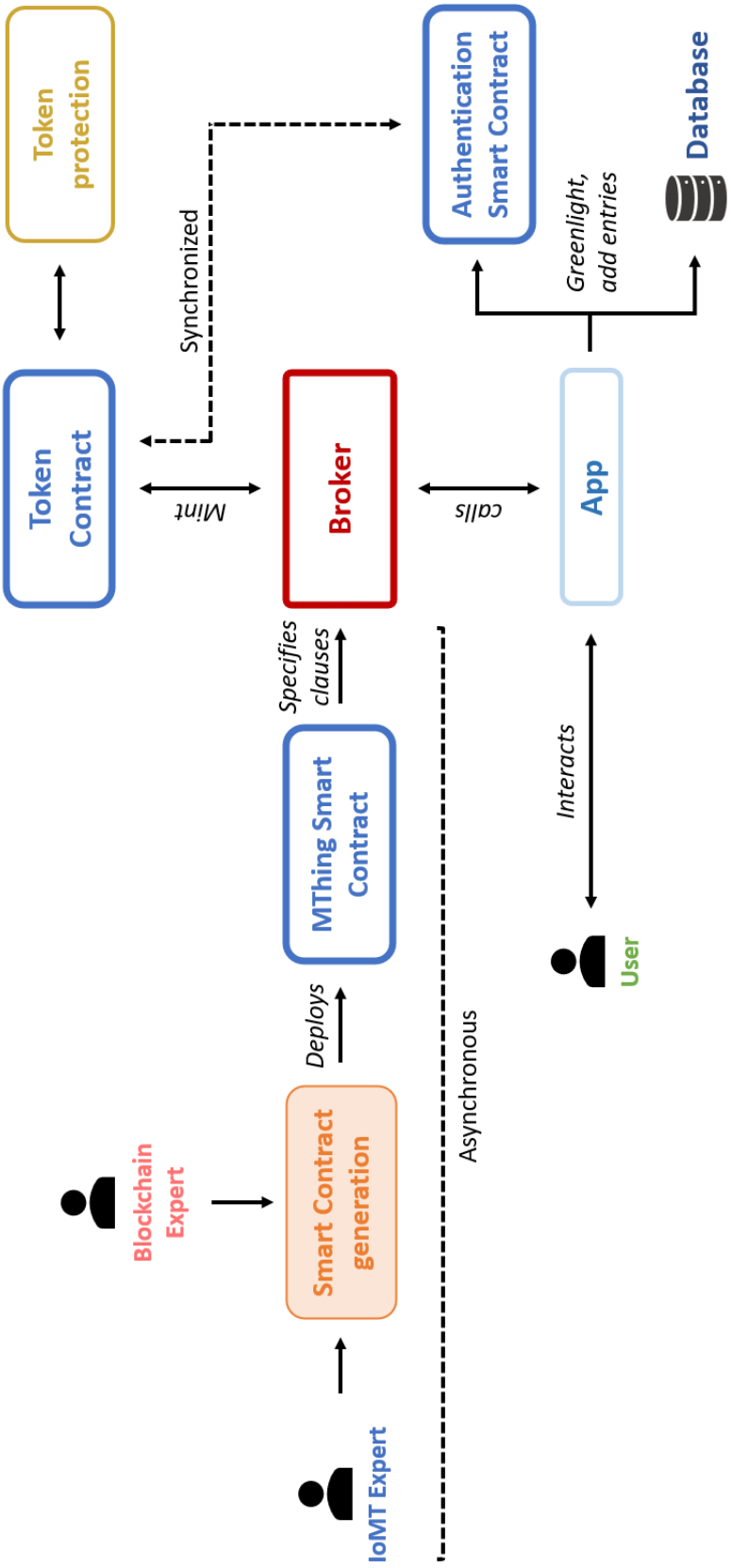

Figure 81: Overview of the complete architecture enabling the complete lifecycle management of Internet of Media Things (IoMT) content. The top left methodological bricks set the MThings up with respect to the Broker, and the right side bricks handle the near-copy verfication and tokenization of accepted entries.

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 initially shown in [ALL21a]. 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 shown in Section III and protected as shown in Section II before being sent to the user as proof of purchase and access to the content. This token's metadata contains information about the MThing in question, the buyer, the purchase of FTs, the time the content was delivered, *etc*. It can further be used as ZKP.

Note that throughout these events, users only interact with the on-chain elements, which ties into the anonymity required by the use case. Off-chain components are only deployed to support functionalities of on-chain software and only stored hashed fingerprinted data, which cannot be connected to personal data. Specifically, we again use the ISCC as a fingerprinting method, as it is suited to the variety of data to be identified via its different code components. As such, the ISCC generated after two buyers purchase the same content at the same time will remain distinct, allowing further use and clearly identifying each party's actions, as required by this use case. To do so, the comparison is made in subparts, *i.e.*, we not only calculate the difference between two given codes but 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.

While the specific means of content distribution is not within the realm of this solution as it is a provision made by MPEG-IoMT, the Broker is informed of the redeeming of the tokens and access to the service when the MThing claims its payment, which launches the minting of the NFT (an ERC721 in this illustration) which is sent to the buyer. The complete sequence of events is shown in Figure 82 and references the prior solutions that power different portions of this combined workflow.

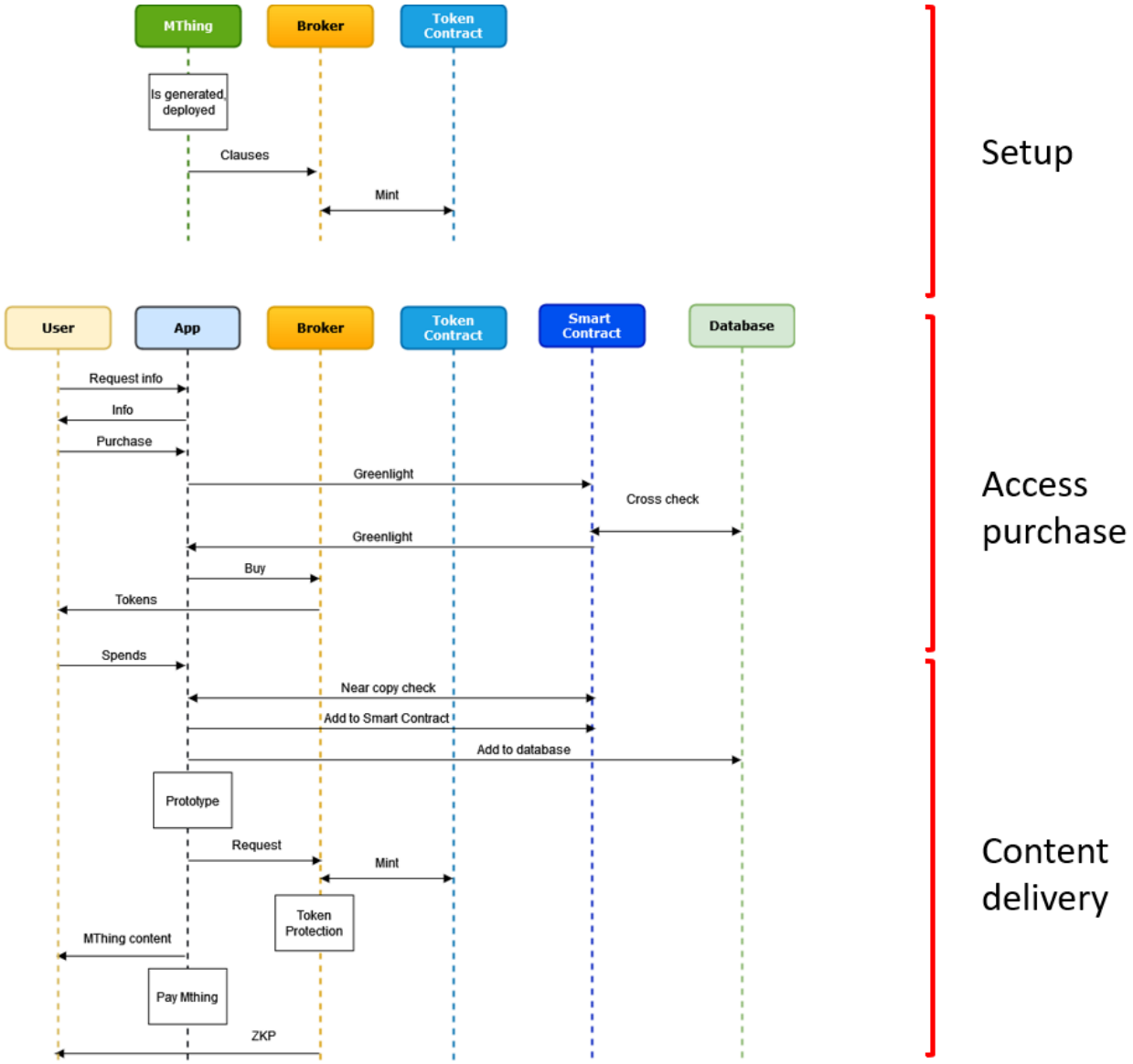

Figure 82: Sequence diagram of the complete lifecycle management of Internet of Media Things (IoMT) content, including the asynchronous setup, the purchase of access tokens, and the delivery of content and Zero-Knowledge Proofs (ZKPs).

At the term of these events:

- The buyer has accessed the content.
- The MThing is paid by the App upon content delivery.
- A protected token is delivered to the buyer as ZKP of the purchase.
- Smart Contract-backed *off-chain* database now records protected instances of the access of the restricted content with unfalsifiable information.

5.V.C. Analysis

This Section provides two use cases with a solution combining methodological bricks introduced in Chapter 4 and illustrated standalone earlier in this chapter. Specifically, this solution combines visual content and blockchain technologies to provide automated, end-to-end lifecycle management of authenticated assets on blockchain environments. As such, the architecture and workflow design are given by 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 component. 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. This can be said with most technologies used in the workflow as not only are the database solution, App programming language, blockchain infrastructure, and token standards are interchangeable, but the very input formats can also be modified to suit specific requirements (*e.g.*, to an audio-oriented use case). Note that 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 the identification of semantic content, which can be put at the service of a wide array of scenarios thanks to the flexibility of the workflow. Yet, 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 replace fingerprinting without affecting the nature of the processing chain. Consequently, our architecture and workflow can seamlessly integrate a wide spectrum of standard visual identification technologies, which we illustrated using very different algorithms in our implementation section. Further media technologies in

the 3D and audio space could also be employed to similar avail or in combination with visual identification, e.g., using MPEG Compact Descriptors for Video Recognition [DUA18].

③ Architectural features: When considering the workflow supported by our architecture, a key element is the complete lifecycle support of assets. The workflow begins multiple steps before creating a blockchain asset and ends with the distribution of 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 when it comes to the interoperable aspect of the software and its components, although we do not deal with an Interoperability Mechanism (IM) as defined by [BEL23]. This is notably ensured throughout the interchangeability and the systematic integration of diverse state-of-the-art methodological bricks from *on* and *off-chain* environments, as mentioned in k. 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 that seamlessly interact 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 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 [GRA90][KAM18]. Lastly, the Token Contract is the only element requiring any extra instantiation in the expansion of 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.

④ Technical challenges: This architecture's central presence of blockchain processing 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. Yet, 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 more efficient *off-chain* technological bricks. Thus, we avoid the brunt of the main issue of *on-chain* development, namely lengthy processing times.

This resource consumption on the blockchain is measured via gas. Yet, gas costs are not only relative to specific blockchains but also to 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. This 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, [WOO14] estimates 640k gas being necessary to store a kilobyte of data on Ethereum's mainnet. As such, a single fingerprint computed with [GAR16], 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 [SMI23b]. 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 are within blockchain standards and participate in the performance preservation discussed in k. Of course, these gas and performance costs relate to actual 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 [ZHA20b] in the applicative blockchain realm in a general effort to provide 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 [ETH23a] 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 [PWC21]. 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.

We proved that our findings can be valuable add-ons to existing solutions, extending their functionalities and automating the process, as shown in the IoMT use case. Hence, future applicative work could enable this solution in light of web3 innovations such as so-called green blockchains [BAD21][SHA20] and energy-conscious blockchain discoveries to ensure the benefit provided by our workflow can come at a minimal cost or even expansive new environments such as the metaverse. From a methodological point of view, future work could not only move further in the exploration of technologies capable of supporting the bricks of the architecture but also make use of emerging soulbound tokens to further the ZKP orientation of the solution or on a shorter time frame mixed token standards enabling the multifaceted reliance on a single Token Contract. Finally, from the perspective of furthering interoperability, future work could formalize IMs provided by this workflow but also use Inter-Blockchain Communication (IBC) [QAS19]-oriented IMs being explored, specifically those enabling token exchanges across infrastructures. Potential additions should remain mindful of retaining the simplicity of the solution.

5.VI. Summary

The implementations presented in this chapter confirm our methodological approach's validity and highlight their most significant strengths and limitations, which are unsurprisingly endemic to blockchain environments. Notably, processing times and gas costs are an issue that cannot be diminished by direct confrontation but by an approach designed to minimize their impact. In the above, we ensured blockchain software only intervened at critical moments of our workflows and treated data prepared so as not to tax resources. This approach, combined with energy-mindful modern blockchain protocols, leads to the heavy dampening of this limitation, leading to low cost and fast processing relative to the environment. Furthermore, blockchain benefits, including transparency and robustness, retain their full effect.

Yet, it is vital to remember that robustness and security are a matter of the weakest link in a chain. It does not matter how hard a safe is to crack if the key is available to everyone. To remedy this issue, we put blockchain data robustness at the service of off-chain databases. Yet, the same logic applies further, placing the onus on the connection to on-chain services. As of the time of writing, no significant vulnerabilities have been found in the libraries used to connect to the most mainstream blockchains, and minor patches are put out regularly. As such, the weakest link in the chain is often the end user and their security awareness. Phishing attacks and private keys stolen or made vulnerable by third-party services are by far the leading causes of data breaches and stolen funds in blockchains.

Our methodological bricks were built with technological flexibility and interoperation at their heart, which led to their joint implementation in Section V. Logically, future work can be carried out from both methodological and technical standpoints to further this approach. This would require the furtherance of features serving specific use cases and the continued application of new blockchain technologies (*e.g.,* soulbound tokens, IBC, *etc*.). Table X provides a succinct summary of our contributions and how they address the current challenges and limitations shown in Table I.

TABLE X
SYNTHESIS OF THE SOLUTIONS BROUGHT FORTH THROUGHOUT CHAPTERS 4 AND 5

Contribution	Innovation	Interoperability front	Future work
1. Brokerage and automatic Smart Contract generation	An on-chain workflow that uses tokens as escrow assets in the provision of a service.	*Intra*-blockchain	Including ZKP in the compensation of providers and creating prototypes compatible with other forms of data structures.
	An off-chain workflow that uses existing descriptive assets (ontologies) and translates them into Smart Contracts.	*Inter*-blockchain	
2. Token Level Smart Contract	A Smart Contract abstracted as a token and serving as fully transparent third party in the application of rules.	*Intra*-blockchain	Integrating the solutions within state-of-the-art interfacing.
3. Smart Contract load balancing	A hybrid architecture that offloads computation to off-chain resources while maintaining on-chain verifiability.	*Extra*-blockchain	Expanding the horizon of compatible operations to be offloaded.
4. Asset lifecycle management for visual content	A multilayered hybrid architecture that makes use of the four above methodologies.	Uses and demonstrates all the above	Implementing further web3 developments and consequently providing new features.

The addresses and testnets of the Smart Contracts used in this chapter can be found in Table XI.

TABLE XI
SUMMARY OF THE ENVIRONMENTS AND ADDRESSES WHERE THE SMART CONTRACTS WE ILLUSTRATED IN THIS CHAPTER WERE DEPLOYED

Solution	Blockchain	Address
IoMT Broker	EVM Hyperledger PoA (link)	0x068c8ACA78816edE02562bdB82DCEaf50A757384
TLSC	EVM Hyperledger PoA (link)	0x2920ef90b2619523acc3b9d74245f133C64b1439
	Tezos Ghostnet	KT1KKTvVMwy9CtjtbV7aXXmKEVoqXwzaKbLB
Automatic Smart Contract generation	Not Applicable[14]	
Load balancing	EVM Hyperledger PoA (link)	0x2b11B01d58114ad2E86FcBdAe68013809F616A71
	Tezos Ghostnet	KT1MSVoHdoYQpWPBXw4QbfvjSU1abizJbzM2
Full lifecycle combination	EVM Hyperledger PoA (link)	0x63eB5fA19eea6199fb81119308950B5dc08fCc7B
	Tezos Ghostnet	KT1MSVoHdoYQpWPBXw4QbfvjSU1abizJbzM2

The rest of the software that was used in these implementations is available at [GTB23b]. The access links to the MPEG reference software we contributed to developing are summarized in Table XII.

TABLE XII
SUMMARY OF THE LINKS TOWARDS THE MPEG REFERENCE SOFTWARE CONTRIBUTED TO IN THE CONTEXT OF THIS THESIS

Smart Contracts for Media	https://standards.iso.org/iso-iec/21000/-23/ed-1/en/
Media Value Chain Ontology	https://standards.iso.org/ittf/PubliclyAvailableStandards/c057394_ISO_IEC_21000-8_2008_Amd_2_2011_Reference_Software.zip
Audio Value Chain Ontology	https://standards.iso.org/iso-iec/21000/-8/ed-2/en/amd/4
Media Contract Ontology	https://standards.iso.org/iso-iec/21000/-21/ed-2/

[14] The workflow produces a Smart Contract, hence a different address, each time it is executed. The reference software linked above can be used to generate and deploy a Smart Contract and will indicate the resulting address.

Chapter 6. Conclusion

This chapter will conclude our work, highlighting our contributions and their place in the state-of-the-art. We shall also use these parting words to gain perspective on the position of blockchains in the landscape of current and future technological advancements.

Media and blockchain are interwoven paradigms with substantial overlaps that promise to grow in the light of the modern technological landscape. The strengths of the state-of-the-art of one complement the other, and vice-versa. Blockchains are a perfect vehicle to carry important media representations into the world of web3 as they support transparent asset exchanges and provide data robustness to unprecedented levels. Conversely, media notions accompanied the applicative era of web3, significantly contributing to the scope of blockchains from a technological niche into one of the most active technological ecosystems, with perspectives growing daily.

Yet, this association has also brought new critical challenges to the table, some of which include (as illustrated in Figure 1):

1. The centralization of asset distribution.
2. The lack of a persistent link between web2 assets and their web3 representations.
3. The high knowledge requirement for developing and deploying Smart Contracts.
4. The computational limitations of Smart Contracts.
5. The difficulty of integrating web3 ideas and concepts into versatile high-abstraction architectures.

We consequently devised methodological solutions to these problems whose processes were abstracted and specified by the identification of design requirements stemming from gaps in the state-of-the-art. As such, we ensured the effectiveness and functional interoperability of each one. Respectively:

1. **Zero-trust brokerage:** Our token distribution solution (cf. Chapter 4 Section II, Chapter 5 Section II) enables for heterogenous protected content to be distributed in a zero-trust fashion with in-built escrow functionalities. It can also distribute access ZKPs for further vertical applications.
2. **Token protection solution:** Our IPR-enforcing Token Level Smart Contract (cf. Chapter 4 Section II, Chapter 5 Section III) uses a shift of blockchain applicative pillars by using Smart Contracts at the same conceptual level as tokens. It builds a strong link between tokens and their web2 equivalents, enabling creators to set rules that apply to their tokens systematically and indefinitely, or until specific conditions are met.
3. **Automatic Smart Contract generation and deployment:** Our workflow (cf. Chapter 4 Section III, Chapter 5 Section II) enables Smart and Token Contracts to be created and deployed on a variety of blockchains with no knowledge requirements past the initial setup. This leaves new applications and use cases to be implemented from formalized human-readable ontologies.
4. **Smart Contract load balancing:** Our approach to furthering Smart Contract computational capacities was to bring forth an architecture allowing for normally prohibitively resource-heavy visual fingerprinting to occur on blockchain environments (cf. Chapter 4 Section III, Chapter 5 Section IV). The method uses Smart Contract storage and cross-environment verification to enable the blockchain authentication of off-chain data entries.
5. **Versatile applicative workflows:** The four above methodological bricks were built with the possibility of versatile applicative associations as a design requirement, facilitating *intra*, *inter*, and *extra*-blockchain interoperability by

eliminating the need for trusted third parties, creating all-but-one-step agnostic processes adaptable to all infrastructures, and permitting new joint uses for web paradigms, respectively. Thus, they can seamlessly connect with one another to tackle new challenging verticals, as illustrated with the full lifecycle management of authenticated assets linked to visual media (cf. Chapter 4 Section IV, Chapter 5 Section V).

We put each of the approaches introduced in this thesis at the service of real-world scenarios before discussing their respective applicative perimeters, promises, shortcomings, and avenues for future work. This analysis raises the generality of the technological innovations at the heart of these solutions, which can be used in a broad spectrum of connected or unrelated applications. Figure 83 shows the evolution of Figure 1 in light of our contributions.

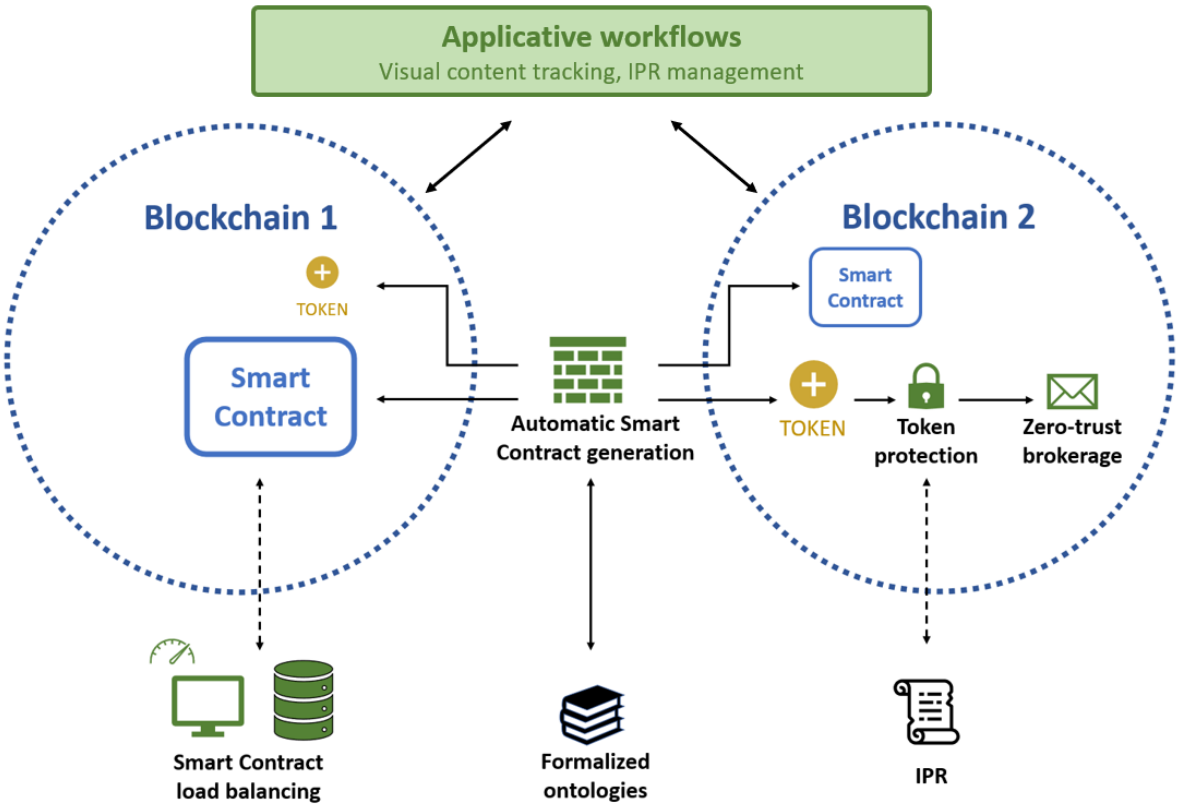

Figure 83: Overview of how the solutions brought forth in this thesis (shown in green) address the applicative blockchain critical limitations identified in our state-of-the-art analysis and summarized in Figure 1. Token-centered solutions were brought forth to enable their protected distribution and persistent Intellectual Property Right links with underlying assets (shown in Blockchain 2). Smart Contract limitations were also tackled by enabling their systematic generation across blockchains and the processing of prior prohibitively complex operations (shown in the center and in Blockchain 1, respectively). Each of these solutions was also built to interoperate with others and further versatile applicative workflows.

The world of blockchain is fast growing and aware of its shortcomings. The challenge in providing solutions to its weaknesses lies in the combination of technical aspects, community-centered questions (*e.g.*, decentralized decision-making, acceptance), and real-world regulations (*e.g.*, taxes, KYC). The technical insights and general understanding

we gathered along our research and contributions to the domain outline three major trends applicative blockchains seem to be following:

1. **Convergence:** Blockchains are evolving towards generalized ecosystems bearing common applicative (*de facto*) standards and seamless data and asset exchanges (*e.g.*, transactions, Smart Contracts, tokens).
2. **Semi trusted hybrid interfacing:** The native high knowledge requirement in engaging with the environment will continue to be compensated by third party interfaces which can technically be verified by their users but scarcely are.
3. **Frontier solution exploitation:** More and more will be done at the frontier between web3 and legacy environments (*e.g.*, layer one solutions), where flexible architectures and workflows are possible. This will not only mitigate some technical shortcomings of blockchains but also enable more effective regulatory interfacing.

These advancements will naturally be slowed down by standardization processes and economic strategies powered by prospective web3 markets but will eventually emerge by virtue of the innate disruptiveness of the space. The time scale of such advancements is difficult to predict given that blockchains live at the intersection of scientific innovation, 10-figure private investments, ideological visions, regulative gray zones, mainstream appeal, and general skepticism. Figure 84 illustrates our perception of what the blockchain space is to become.

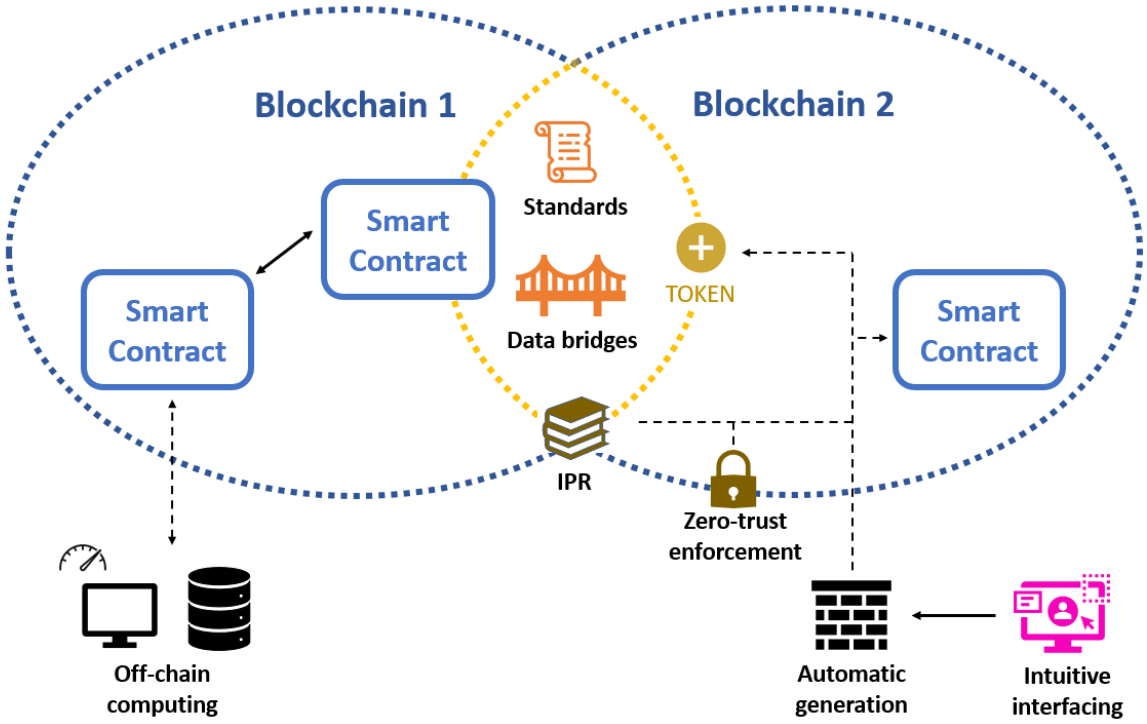

Figure 84: Overview of our perception of the direction applicative blockchains are taken. The three major trends we identified are: convergence (shown in the intersection of Blockchain 1 and Blockchain 2 and via the existence of common standards, in orange), semi-trusted hybrid interfacing (shown at the bottom right in the context of Smart Contract and Token creation, in magenta), and frontier solution exploitation (shown in the interoperation of off-chain and on-chain technologies and in the existence of solutions living between the two realms, in brown)

If our projections happen to materialize, the most significant shift regarding the technology would lie in the generalized public trust in blockchains. This trust would translate to new initiatives and more traffic resulting in further resources invested in solutions that will become easier to use and widely interfaced. Down this line of thinking, blockchains could eventually become a flexible tool perfectly adapted to transversal applications. At the time of writing, blockchains are predominantly perceived as niche payment systems by most. Yet, the notion of value exchange is so robust and anchored in our daily lives that it is hard to envision large-scale disruption within its bounds. Thus, further blockchain (and, more generally, web3) disruption must happen at the frontiers of value exchange and many more known fields, bridging the gap between paradigms we perceive as separate. Prospective horizons could span payment systems, communication, entertainment, property, digital identity, and many more. Time will tell.

Despite these prospects, it is key to retain perspective on the main features and pitfalls of blockchains and web3 solutions. Indeed, technological trends pushed by financial institutions looking for lucrative opportunities and news outlets battling for readers' attention have sometimes gotten in the way of thoughtful applications and scientific outlooks. Throughout the years leading up to this thesis, blockchains have seen billions of dollars in questionable investments pushed by their previous successes. As summarized by [COI20b]:

There's a market for magic, and that market is big.

Web3 solutions were praised as an end-all-be-all solution to all of humanity's problems and pointed out as useless technological toys for people and businesses with too much money on their hands, sometimes simultaneously. Of course, such phenomena are common when emerging technologies hit mainstream attention, but the awareness around blockchains was further bolstered by a worldwide pandemic raising novel challenges to be tackled and freeing up the attention of many. Fortunately, time tends to filter out excessive and unjustified uses and will eventually only leave applications that genuinely benefit from blockchains and mitigate their shortcomings.

Although the ideas in this thesis cannot wholly solve the looming issues blockchains must face, they contribute unique and innovative approaches, inviting new work and eventual robust standards to emerge. The existence of significant loopholes and malicious actors in the blockchain space should not be faced with fatalism but with critical realism. Further technical and methodological advancements contribute to the establishment of an adequately explored and documented field. Each step towards an interoperable standard environment fosters trust between actors, feeding a virtuous cycle of new traffic and better standards.

References

[ABB22] Abbas, H., & Di Pietro, R. (2022). *Sanitization of Visual Multimedia Content: A Survey of Techniques, Attacks, and Future Directions*. arXiv preprint arXiv:2207.02051.

[ABH19] Abhishta, A., Joosten, R., Dragomiretskiy, S., & Nieuwenhuis, L. J. (2019). *Impact of successful ddos attacks on a major crypto-currency exchange*. In 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP) (pp. 379-384). IEEE.

[AGO23] Agora Developer platform. Available: https://docs.agora.io/en/

[ALB20] Al-Breiki, H., Rehman, M. H. U., Salah, K., & Svetinovic, D. (2020). *Trustworthy blockchain oracles: review, comparison, and open research challenges*. *IEEE access*, *8*, 85675-85685.

[ALC23a] *Web3 Social Media dApps*. Alchemy. Available: https://www.alchemy.com/best/web3-social-media-dapps (accessed September 2023).

[ALC23b] *Web3 Games.* Alchemy. https://www.alchemy.com/best/web3-games (accessed September 2023).

[ALL19] Allombert, V., Bourgoin, M., & Tesson, J. (2019). *Introduction to the tezos blockchain*. In 2019 International Conference on High Performance Computing & Simulation (HPCS) (pp. 1-10). IEEE.

[ALL21] Allouche, M., Frikha, T., Mitrea, M., Memmi, G., & Chaabane, F. (2021). *Lightweight blockchain processing. Case study: scanned document tracking on tezos blockchain*. Applied Sciences, 11(15), 7169.

[ALL21a] Allouche, M., Mitrea, M., Moreaux, A., & Kim, S. K. (2021). *Automatic smart contract generation for Internet of media things*. ICT Express, 7(3), 274-277.

[ALL21b] Allouche, M., Ljubojevic, M., & Mitrea, M. (2021). *Visual document tracking and blockchain technologies in mobile world*. Electronic Imaging, 2021(8), 279-1.

[ALL22] Allouche, M., & Mitrea, M. (2022). *Video fingerprinting: Past, present, and future*. Frontiers in Signal Processing, 2, 984169.

[AMI19] Amirault, R. J. (2019). *The next great educational technology debate: Personal data, its ownership, and privacy*. Quarterly Review of Distance Education, 20(2), 55-73.

[ANG17] Angraal, S., Krumholz, H. M., & Schulz, W. L. (2017). *Blockchain technology: applications in health care*. Circulation: Cardiovascular quality and outcomes, 10(9), e003800.

[ANJ17] Anjum, A., Sporny, M., & Sill, A. (2017). *Blockchain standards for compliance and trust*. IEEE Cloud Computing, 4(4), 84-90.

[APP22] Appelbaum, D., Cohen, E., Kinory, E., & Stein Smith, S. (2022). *Impediments to blockchain adoption*. Journal of Emerging Technologies in Accounting, 19(2), 199-210.

[ARC03] Arce, I. (2003). *The weakest link revisited [information security]*. IEEE Security & Privacy, 1(2), 72-76.

[AXA23] *AXA se lance sur la Blockchain avec fizzy*. (2017) AXA. Available : https://www.axa.com/fr/actualites/axa-se-lance-sur-la-blockchain-avec-fizzy (accessed September 2023).

[AZB21] Azbeg, K., Ouchetto, O., Jai Andaloussi, S., & Fetjah, L. (2021). *An overview of blockchain consensus algorithms: Comparison, challenges and future directions*. Advances on Smart and Soft Computing: Proceedings of ICACIn 2020, 357-369.

[BAC18] Bach, L. M., Mihaljevic, B., & Zagar, M. (2018). *Comparative analysis of blockchain consensus algorithms*. In 2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO) (pp. 1545-1550). IEEE.

[BAD21] Bada, A. O., Damianou, A., Angelopoulos, C. M., & Katos, V. (2021). *Towards a green blockchain: A review of consensus mechanisms and their energy consumption*. In 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS) (pp. 503-511). IEEE.

[BAK21] Bakare, L. *Collector buys fake Banksy NFT for £244,000*. (2021) The Guardian. Available: https://www.theguardian.com/technology/2021/sep/01/collector-buys-fake-banksy-nft-for-244000 (accessed September 2023).

[BAM20] Bamakan, S. M. H., Motavali, A., & Bondarti, A. B. (2020). *A survey of blockchain consensus algorithms performance evaluation criteria*. Expert Systems with Applications, 154, 113385.

[BAM22] Bamakan, S. M. H., Nezhadsistani, N., Bodaghi, O., & Qu, Q. (2022). *Patents and intellectual property assets as non-fungible tokens; key technologies and challenges*. Scientific Reports, 12(1), 2178.

[BAT13] Batten, L. M. (2013). *Public key cryptography: applications and attacks*. John Wiley & Sons.

[BAY93] Bayer, D., Haber, S., & Stornetta, W. S. (1993). *Improving the efficiency and reliability of digital time-stamping*. In Sequences II: Methods in Communication, Security, and Computer Science (pp. 329-334). Springer New York.

[BEL21] Belchior, R., Vasconcelos, A., Guerreiro, S., & Correia, M. (2021). *A survey on blockchain interoperability: Past, present, and future trends*. ACM Computing Surveys (CSUR), 54(8), 1-41.

[BEL23] Belchior, R., Riley, L., Hardjono, T., Vasconcelos, A., & Correia, M. (2023). *Do you need a distributed ledger technology interoperability solution?*. Distributed Ledger Technologies: Research and Practice, 2(1), 1-37.

[BEL94] Bellare, M., Kilian, J., & Rogaway, P. (1994, August). *The security of cipher block chaining*. In Annual International Cryptology Conference (pp. 341-358). Berlin, Heidelberg: Springer Berlin Heidelberg.

[BLA19] Black, M., Liu, T., & Cai, T. (2019). *Atomic loans: Cryptocurrency debt instruments*. arXiv preprint arXiv:1901.05117.

[BLO23] *Ethereum NFT Marketplace Monthly Volume*. (2023) The Block. Available : https://www.theblock.co/data/nft-non-fungible-tokens/marketplaces/nft-marketplace-monthly-volume (accessed September 2023).

[BOO23] *Bootstrap Developer Portal.* Available : https://getbootstrap.com/docs/4.1/getting-started/introduction/ (accessed September 2023).

[BRC23] *Tokenization Global Market Report 2023*. (2023) The Business Research Company. Available: https://www.thebusinessresearchcompany.com/report/tokenization-global-market-report (accessed September 2023).

[BRE01] Brewer, E. A. (2001). *Lessons from giant-scale services*. IEEE Internet computing, 5(4), 46-55.

[BRE12] Brewer, E. (2012). *CAP twelve years later: How the" rules" have changed*. Computer, 45(2), 23-29.

[BRE21] Breidenbach, L., Cachin, C., Chan, B., Coventry, A., Ellis, S., Juels, A., ... & Zhang, F. (2021). *Chainlink 2.0: Next steps in the evolution of decentralized oracle networks*. Chainlink Labs, 1, 1-136.

[BRO18] Broncker, J., & Veuger, J. (2018). *Blockchain: technology looking for a problem? Visions on the application of blockchain technology in real estate*. Barometer Public Real Estate, 122.

[BTC13] *11/12 March 2013 Chain Fork Information*. (2013) Bitcoin. Available: https://bitcoin.org/en/alert/2013-03-11-chain-fork (accessed September 2023).

[BUC16] Buchman, Ethan. Tendermint: Byzantine fault tolerance in the age of blockchains. Diss. University of Guelph, 2016.

[BUC21] Buck, C., Olenberger, C., Schweizer, A., Völter, F., & Eymann, T. (2021). *Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust*. Computers & Security, 110, 102436.

[BUH23] Buhalis, D., Lin, M. S., & Leung, D. (2022). *Metaverse as a driver for customer experience and value co-creation: implications for hospitality and tourism management and marketing*. International Journal of Contemporary Hospitality Management, 35(2), 701-716.

[BUR20] Burks, Z., Morgan, J., Malone, B., Seibel, J. *ERC-2981: NFT Royalty Standard*. (2020) Ethereum Improvement Proposals. Available: https://eips.ethereum.org/EIPS/eip-2981. (accessed September 2023).

[BUT13] Buterin, V. (2013) *Selfish Mining: A 25 attack against the bitcoin network*. Bitcoin Magazine. Available : https://bitcoinmagazine.com/technical/selfish-mining-a-25-attack-against-the-bitcoin-network-1383578440 (accessed September 2023).

[BUT14] Buterin, V. (2014). *A next-generation smart contract and decentralized application platform*. White paper, 3(37), 2-1.

[BWC22] *Global Non-Fungible Token (NFT) Market Size to reach USD 20 billion by 2028*. (2022) BlueWeave Consulting. Available: https://www.globenewswire.com/news-release/2022/09/12/2514295/0/en/Global-Non-Fungible-Token-NFT-Market-Size-to-Reach-USD-20-billion-by-2028-BlueWeave-Consulting.html (accessed September 2023).

[CAL20] Caldarelli, G. (2020). *Understanding the blockchain oracle problem: A call for action*. Information, 11(11), 509.

[CAO11] Cao, L., Qi, G., Tsai, S. F., Tsai, M. H., Pozo, A. D., Huang, T. S., ... & Lim, S. H. (2011). *Multimedia information networks in social media*. Social Network Data Analytics, 413-445.

[CAS21] Castellon, C., Roy, S., Kreidl, P., Dutta, A., & Bölöni, L. (2021). *Energy efficient merkle trees for blockchains*. In 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 1093-1099). IEEE.

[CCO23] *Attribution 4.0 International*. Creative Commons. Available: https://creativecommons.org/licenses/by/4.0/legalcode (accessed September 2023).

[CHA22a] Chang, E. C., & Tai, N. C. (2022, October). *Non-Fungible Token Donation Platform Delivers Continuous Goodwill without Daunting Donor Commitment*. In 2022 IEEE 11th Global Conference on Consumer Electronics (GCCE) (pp. 852-853). IEEE.

[CHA22b] *The Chainalysis 2021 NFT Market Report. (2022)*.Chainalysis. Available : https://nyko.io/wp-content/uploads/2022/02/Chainalysis-NFT-Market-Report.pdf (accessed September 2023).

[CHA22c] *DeFi Takes on Bigger Role in Money Laundering But Small Group of Centralized Services Still Dominate*. (2022) Available: https://www.chainalysis.com/blog/2022-crypto-crime-report-preview-cryptocurrency-money-laundering/ (accessed September 2023).

[CHAI23] *What is a blockchain oracle?*. (2023) Chainlink. Available: https://chain.link/education/blockchain-oracles (accessed September 2023).

[CHD16] *Global Blockchain Forum Launched to Build Interoperable Public Policy for $8Bil Industry* [...] . (2016) Global Blockchain Forum. Available: https://digitalchamber.org/global-blockchain-forum-launched/ (accessed September 2023).

[CHE19] Chen, H., Rouhani, B. D., Fu, C., Zhao, J., & Koushanfar, F. (2019). *Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models*. In Proceedings of the 2019 on International Conference on Multimedia Retrieval (pp. 105-113).

[CHE19a] Chen, S. C. (2019). *Multimedia for autonomous driving*. IEEE MultiMedia, 26(3), 5-8.

[CHE20] Chen, Y., & Bellavitis, C. (2020). *Blockchain disruption and decentralized finance: The rise of decentralized business models*. Journal of Business Venturing Insights, 13, e00151.

[CHE22] Chen, S. C. (2022). *Multimedia research toward the metaverse*. IEEE MultiMedia, 29(1), 125-127.

[CHO18] Choudhury, O., Rudolph, N., Sylla, I., Fairoza, N., & Das, A. (2018). *Auto-generation of smart contracts from domain-specific ontologies and semantic rules*. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (pp. 963-970). IEEE.

[CHR19] Christman, E. (2019) *Just How Much Money is There in Unclaimed Black Box Royalties?*. Available: https://www.billboard.com/pro/unclaimed-black-box-royalties-how-much-money/ (accessed September 2023).

[CIS20] Cisco, U. (2020). *Cisco annual Internet report (2018–2023)* White paper. Cisco: San Jose, CA, USA, 10(1), 1-35.

[CMC23] *Tezos price fluctuations*. CoinMarketCap. Available: https://coinmarketcap.com/currencies/tezos/ (accessed September 2023).

[COI20a] *Chinese Authorities Have Seized a Massive $4B in Crypto From PlusToken Scam. (2020) CoinDesk. Available* : https://www.coindesk.com/markets/2020/11/27/chinese-authorities-have-seized-a-massive-4b-in-crypto-from-plustoken-scam/ (accessed September 2023).

[COI20b] *Blockchain, the amazing solution for almost nothing*. (2020) The Correspondent. Available: https://thecorrespondent.com/655/blockchain-the-amazing-solution-for-almost-nothing (accessed September 2023).

[COI23] Coinbase. Available: https://nft.coinbase.com/ (accessed September 2023).

[COL19] Cole, R., Stevenson, M., & Aitken, J. (2019). *Blockchain technology: implications for operations and supply chain management*. Supply Chain Management: An International Journal, 24(4), 469-483.

[CON04] Conti, G., & Abdullah, K. (2004). *Passive visual fingerprinting of network attack tools*. In Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security (pp. 45-54).

[CON21] *An analysis of Ethereum's decentralized finance ecosystem in Q2 2021*. (2021) Consensys. Available : https://consensys.net/reports/defi-report-q2-2021 (accessed September 2023).

[COX07] Cox, A. (2007). *Flickr: What is new in Web2. 0*. Towards a social science of Web, 2.

[CRY22] *How a Block in the Bitcoin Blockchain Works*. (2022) Cryptopedia. Available: https://www.gemini.com/cryptopedia/what-is-block-in-blockchain-bitcoin-block-size (accessed September 2023).

[CRY23] CryptoKitties. Available: https://www.cryptokitties.co/ (accessed September 2023).

[CSI23] *Escrow*, CSIRO Blockchain Patterns. Available: https://research.csiro.au/blockchainpatterns/general-patterns/blockchain-payment-patterns/escrow-2/ (accessed September 2023).

[DAP23a] *State of Blockchain Gaming in Q1 2023*. (2023) DappRadar. Available: https://dappradar.com/blog/state-of-blockchain-gaming-in-q1-2023 (accessed September 2023).

[DAP23b] *Best NFT Marketplaces*. DappRadar. Available: https://dappradar.com/rankings/nft/marketplaces (accessed September 2023).

[DAS22] Das, D., Bose, P., Ruaro, N., Kruegel, C., & Vigna, G. (2022). *Understanding security issues in the NFT ecosystem*. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (pp. 667-681).

[DAT23] *Piracy Is Back: Piracy Statistics for 2023*. (2023) DataProt. Available: https://dataprot.net/statistics/piracy-statistics/ (accessed September 2023).

[DAV18] Davis, S., Arslanian, H., Fong, D., Watkins, A., Gee, W., & Cheung, C. Y. (2018). *PwC's Global Blockchain Survey 2018*.

[DHI22] Dhillon, V., Metcalf, D., & Hooper, M. (2017). *Blockchain enabled applications*. Berkeley, CA: Apress.

[DOJ22a] *BitConnect Founder Indicted in Global $2.4 Billion Cryptocurrency Scheme*. (2022) Office of Public Affairs, U.S. Department of Justice. Available: https://www.justice.gov/opa/pr/bitconnect-founder-indicted-global-24-billion-cryptocurrency-scheme (accessed September 2023).

[DOJ22b] *United States Attorney Announces Charges Against FTX Founder Samuel Bankman-Fried*. (2022) Office of Public Affairs, U.S. Department of Justice. Available: https://www.justice.gov/usao-sdny/pr/united-states-attorney-announces-charges-against-ftx-founder-samuel-bankman-fried (accessed September 2023).

[DOJ23] *Co-Founder Of Multibillion-Dollar Cryptocurrency Scheme "OneCoin" Sentenced To 20 Years In Prison*. (2023) Office of Public Affairs, U.S. Department of Justice. Available: https://www.justice.gov/usao-sdny/pr/co-founder-multibillion-dollar-cryptocurrency-scheme-onecoin-sentenced-20-years-prison (accessed September 2023).

[DON22] Dong, H., & Lee, J. S. (2022). *The metaverse from a multimedia communications perspective*. IEEE MultiMedia, 29(4), 123-127.

[DON92] Dony, C., Malenfant, J., & Cointe, P. (1992). *Prototype-based languages: from a new taxonomy to constructive proposals and their validation*. In Conference proceedings on Object-oriented programming systems, languages, and applications (pp. 201-217).

[DON98] Dony, C., Malenfant, J., & Bardou, D. (1998). *Classifying prototype-based programming languages*. Prototype-based Programming: Concepts, Languages and Applications, 86, 71.

[DRI03] Driscoll, K., Hall, B., Sivencrona, H., Zumsteg, P. (2023). *Byzantine fault tolerance, from theory to reality*. International Conference on Computer Safety, Reliability, and Security. Berlin, Heidelberg: Springer Berlin Heidelberg.

[DUA18] Duan, L. Y., Lou, Y., Bai, Y., Huang, T., Gao, W., Chandrasekhar, V., ... & Kot, A. C. (2018). *Compact descriptors for video analysis: The emerging MPEG standard*. IEEE MultiMedia, 26(2), 44-54.

[DUA21] Duan, H., Li, J., Fan, S., Lin, Z., Wu, X., & Cai, W. (2021). *Metaverse for social good: A university campus prototype*. In Proceedings of the 29th ACM international conference on multimedia (pp. 153-161).

[EEA23] *Enterprise Ethereum Alliance Portal*. Available: https://entethalliance.org/ (accessed September 2023).

[EIP23] *ERC*. Ethereum Improvements Proposals. Available: https://eips.ethereum.org/erc (accessed September 2023).

[EMR23] *Tokenization Market, By Component (Solutions and Services), By Organization Size* […]. (2023) EMERGEN Research. Available: https://www.emergenresearch.com/industry-report/tokenization-market (accessed September 2023).

[ENT18] Entriken, W., Shirley, D., Evans, J., Sachs, N. *EIP-721: NonFungible Token Standard*. (2018) Ethereum Improvement Proposals. Available: https://eips.ethereum.org/EIPS/eip-721 (accessed September 2023).

[ETH23a] *Ethereum's energy expenditure*. (2023) Ethereum. Available: https://ethereum.org/en/energy-consumption/#proof-of-stake-energy (accessed September 2023).

[ETH23b] *Middleware*. Web3py Documentation. Available : https://web3py.readthedocs.io/en/stable/middleware.html (accessed September 2023).

[EVA19] Evans, T. M. (2019). *Cryptokitties, cryptography, and copyright*. AIPLA QJ, 47, 219.

[EYA18] Eyal, I., & Sirer, E. G. (2018). *Majority is not enough: Bitcoin mining is vulnerable*. Communications of the ACM, 61(7), 95-102.

[EYS07] Eysenbach, G. (2007). *From intermediation to disintermediation and apomediation: new models for consumers to access and assess the credibility of health information in the age of Web2. 0*. Studies in health technology and informatics, 129(1), 162.

[EZZ22] Ezzat, S. K., Saleh, Y. N., & Abdel-Hamid, A. A. (2022). *Blockchain oracles: State-of-the-art and research directions*. IEEE Access, 10, 67551-67572.

[FBI23] *Blockchain Technology Market Size, Share & COVID-19 Impact Analysis* […]. (2023) Fortune Business Insights. Available: https://www.fortunebusinessinsights.com/industry-reports/blockchain-market-100072 (accessed September 2023).

[FIT03] Fitzpatrick, S. M. (2003). *Stones of the Butterfly: An Archaeological Investigation of Yapese Stone Money Quarries in Palau, Western Caroline Islands, Micronesia*. University of Oregon.

[FRA16] Frantz, C. K., & Nowostawski, M. (2016). From institutions to code: Towards automated generation of smart contracts. In 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W) (pp. 210-215). IEEE.

[FRA20] Frattolillo, F. (2020). *A watermarking protocol based on Blockchain*. Applied Sciences, 10(21), 7746.

[FRA22] Frassetto, T., Jauernig, P., Koisser, D., Kretzler, D., Schlosser, B., Faust, S., & Sadeghi, A. R. (2022). *POSE: Practical off-chain smart contract execution*. arXiv preprint arXiv:2210.07110.

[FRI00] Fridrich, J., & Goljan, M. (2000). *Robust hash functions for digital watermarking*. Proceedings International Conference on Information Technology: Coding and Computing , pp. 178-183. IEEE.

[GAI17] Gainsbury, S. M., & Blaszczynski, A. (2017). *How blockchain and cryptocurrency technology could revolutionize online gambling*. Gaming Law Review, 21(7), 482-492.

[GAN23] Gangwal, A., Gangavalli, H. R., & Thirupathi, A. (2023*). A survey of Layer-two blockchain protocols*. Journal of Network and Computer Applications, 209, 103539.

[GAR16] Garboan, A., & Mitrea, M. (2016). *Live camera recording robust video fingerprinting*. Multimedia Systems, 22, 229-243.

[GFI23] *Ether to euro*. Google Finance. Available: https://www.google.com/finance/quote/ETH-EUR (accessed September 2023).

[GOL94] Goldreich, O., & Oren, Y. (1994). *Definitions and properties of zero-knowledge proof systems*. Journal of Cryptology, 7(1), 1-32.

[GRA23] *A picture is worth a thousand words*. Grammarist. Available : https://grammarist.com/proverb/a-picture-is-worth-a-thousand-words/ (accessed September 2023).

[GRA90] Graefe, G., & Shapiro, L. D. (1990). *Data compression and database performance*. University of Colorado, Boulder, Department of Computer Science.

[GTB20] *Discussion for EIP-2981: NFT Royalty Standard*. (2020) Github. Available: https://github.com/ethereum/EIPs/issues/2907 (accessed September 2023).

[GTB22a] *Ethereum smart contracts (truffle) for marketplace*. (2022) Github. Available: https://github.com/streamr-dev/marketplace-contracts/ (accessed September 2023).

[GTB22b] *Alethio's Light Weight Open Source Ethereum Explorer*. (2022) Github. Available: https://github.com/Alethio/ethereum-lite-explorer (accessed September 2023).

[GTB23a] *IERC721Receiver*. (2023) Github. Available: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/IERC721Receiver.sol (accessed September 2023).

[GTB23b] *Thesis author's GitHub*. (2023) Github. Available: https://github.com/a-moreaux (accessed September 2023).

[GVR23a] *Blockchain Technology Market Size, Share & Trends Analysis* […]. (2023) Grand View Research. Available: https://www.grandviewresearch.com/industry-analysis/blockchain-technology-market (accessed September 2023).

[GVR23b] *Digital Content Creation Market Size, Share & Trends Analysis* […]. (2023) Grand View Research. Available: https://www.grandviewresearch.com/industry-analysis/digital-content-creation-market-report (accessed September 2023).

[HAB91] Haber, S., & Stornetta, W. S. (1991). *How to time-stamp a digital document* (pp. 437-455). Springer Berlin Heidelberg.

[HAD12] Hadmi, A., Puech, W., Said, B. A. E., & Ouahman, A. A. (2012). *Perceptual image hashing*. Watermarking-Volume 2. IntechOpen.

[HAM16a] Hamlin, A., Schear, N., Shen, E., Varia, M., Yakoubov, S., & Yerukhimovich, A. (2016). *Cryptography for big data security* (pp. 241-288). Taylor & Francis LLC, CRC Press.

[HAM16b] Hampton, N. *Understanding the blockchain hype: Why much of it is nothing more than snake oil and spin.* (2016) Computerworld. Available: https://www2.computerworld.com.au/article/606253/understanding-blockchain-hype-why-much-it-nothing-more-than-snake-oil-spin/ (accessed September 2023).

[HE16] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[HE20] He, D., Deng, Z., Zhang, Y., Chan, S., Cheng, Y., & Guizani, N. (2020). *Smart contract vulnerability analysis and security audit*. IEEE Network, 34(5), 276-282.

[HED18] *Hedera: A Public Hashgraph Network & Governing Council*. (2018) Hedera Hashgraph whitepaper.

[HED23] *Gossip about gossip*. Hedera. Available: https://docs.hedera.com/hedera/core-concepts/hashgraph-consensus-algorithms/gossip-about-gossip (accessed September 2023).

[HEI21] Heines, R., Dick, C., Pohle, C., & Jung, R. (2021). *The Tokenization of Everything: Towards a Framework for Understanding the Potentials of Tokenized Assets*. In PACIS (p. 40).

[HEL21] Hellani, H., Sliman, L., Samhat, A. E., & Exposito, E. (2021). *Tangle the blockchain: towards connecting blockchain and DAG*. IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 63-68). IEEE.

[HEN23] Henrickson, E. *The Cyber Security of Blockchain Networks*. Bemidji State University.

[HEW21] Hewa, T. M., Hu, Y., Liyanage, M., Kanhare, S. S., & Ylianttila, M. (2021). *Survey on blockchain-based smart contracts: Technical aspects and future research*. IEEE Access, 9, 87643-87662.

[HOL17] *HOLO Green Paper*. Holo (2017).

[HOL23]*The DHT: A Shared, Distributed Graph Database*. Holochain developer portal. Available: https://developer.holochain.org/concepts/4_dht/ (accessed September 2023).

[HU21] Hu, Q., Yan, B., Han, Y., & Yu, J. (2021). *An improved delegated proof of stake consensus algorithm*. Procedia Computer Science, 187, 341-346.

[I3E23] *IEEE Portal*. IEEE. Available: https://www.ieee.org/ (accessed September 2023).

[INA23] *INATBA Portal*. INATBA. Available: https://inatba.org/ (accessed September 2023).

[IOT22] *Permissionless Innovation*. Iota for business whitepaper.

[ISC23a] *ISCC Foundation Portal*. ISCC. Available: https://iscc.foundation/iscc/ (accessed September 2023).

[ISC23b] *The International Standard Content Code (ISCC)*. Liccium. Available: https://docs.liccium.com/whitepaper/declarations/the-international-standard-content-code-iscc (accessed September 2023).

[ISO16a] *ISO/IEC 21000-20 Information technology — Multimedia framework (MPEG-21) — Part 20: Contract Expression Language.* (2016) ISO.

[ISO16b] *ISO/TC 307 Blockchain and distributed ledger technologies*. (2016) ISO.

[ISO17] *ISO/IEC 21000-21 Information technology — Multimedia framework (MPEG-21) — Part 21: Media contract ontology* (2017) ISO.

[ISO21a] *ISO/IEC 23093-1, Information technology – Internet of media things – Part 1: Architecture*. (2021) ISO.

[ISO21b] *ISO/IEC 23093-2, Information technology – Internet of media things – Part 2: Discovery and communication API*. (2021) ISO.

[ISO22a] *ISO/IEC 21000-23 Smart Contracts for Media*. (2022) ISO.

[ISO22b] *ISO/IEC 23093-3:2022 Information technology — Internet of media things — Part 3: Media data formats and APIs*. (2022) ISO.

[ISO23a] *ISO/IEC JTC1 SC29 WG7 m56823 Advanced usage of DLT solutions for MCameras*. (2021) ISO.

[ISO23b] *ISO/IEC JTC1 SC29 WG7 m63324 IoMT NFT*. (2023) ISO.

[JAC16] Jacobovitz, O. (2016). *Blockchain for identity management*. The Lynne and William Frankel Center for Computer Science Department of Computer Science. Ben-Gurion University, Beer Sheva, 1, 9.

[JAN10] Jang, I., Kudumakis, P., Sandler, M., & Kang, K. (2010). *The MPEG interactive music application format standard [standards in a nutshell]*. IEEE Signal Processing Magazine, 28(1), 150-154.

[JIA16] Jiang, Y. G., & Wang, J. (2016). Partial copy detection in videos: *A benchmark and an evaluation of popular methods*. IEEE Transactions on Big Data, 2(1), 32-42.

[JIN21] Jin, J., Zhang, X., Fu, X., Zhang, H., Lin, W., Lou, J., & Zhao, Y. (2021). *Just noticeable difference for deep machine vision*. IEEE Transactions on Circuits and Systems for Video Technology, 32(6), 3452-3461.

[KAM18] Kamatkar, S. J., Kamble, A., Viloria, A., Hernández-Fernandez, L., & Cali, E. G. (2018). *Database performance tuning and query optimization. In Data Mining and Big Data*: Third International Conference, DMBD 2018, Shanghai, China, June 17–22, 2018, Proceedings 3 (pp. 3-11). Springer International Publishing.

[KAT10] Katz, J. (2010). *Digital signatures (Vol. 1)*. Berlin: Springer.

[KAZ21] Kazmi, A. R., Afzal, M., Abbas, H., Tahir, S., & Rauf, A. (2021, October). *Is Blockchain Overrated?*. In 2021 IEEE 19th International Conference on Embedded and Ubiquitous Computing (EUC) (pp. 187-192). IEEE.

[KA21a] Khalil, M., Khawaja, K. F., & Sarfraz, M. (2021). *The adoption of blockchain technology in the financial sector during the era of fourth industrial revolution: a moderated mediated model*. Quality & Quantity, 1-18.

[KHA21b] Khan, S., Amin, M. B., Azar, A. T., & Aslam, S. (2021). *Towards interoperable blockchains: A survey on the role of smart contracts in blockchain interoperability*. IEEE Access, 9, 116672-116691.

[KIR17] Kirit, N., & Sarkar, P. (2017). *EscrowChain: Leveraging ethereum blockchain as escrow in real estate*. International Journal of Innovative Research in Computer and Communication Engineering, 5(10).

[KOR17] Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., & Kompatsiaris, Y. (2017). *Near-duplicate video retrieval by aggregating intermediate cnn layers*. In MultiMedia Modeling: 23rd International Conference, MMM 2017, Reykjavik, Iceland, January 4-6, 2017, Proceedings, Part I 23 (pp. 251-263). Springer International Publishing.

[KOS20] Koschmann, A., & Qian, Y. (2020). Latent Estimation of Piracy Quality and Its Effect on Revenues and Distribution: The Case of Motion Pictures (No. w27649). National Bureau of Economic Research.

[KRI12] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks*. Advances in neural information processing systems, 25.

[KSH22] Kshetri, N. (2022). *Scams, frauds, and crimes in the nonfungible token market*. Computer, 55(4), 60-64.

[LAR14] Larimer, D. (2014). Delegated proof-of-stake (dpos). *Bitshare whitepaper*, *81*, 85.

[LAU99] Laurence, S., & Margolis, E. (1999). *Concepts and cognitive science*.

[LAV22] Lavaur, T., Lacan, J., & Chanel, C. P. (2022). *Enabling blockchain services for IoE with Zk-Rollups*. Sensors, 22(17), 6493.

[LAW23] *Personal Use License definition* . Law Insider. Available: https://www.lawinsider.com/dictionary/personal-use-license (accessed September 2023).

[LEA19] Walker, G. (2019) *Longest Chain*. Learn me a bitcoin. Available : https://learnmeabitcoin.com/technical/longest-chain (accessed September 2023).

[LED22] Ledesma, L. Vitalik Buterin Discusses Ethereum's Upcoming 'Merge' and 'Surge' at EthCC in Paris. (2022) CoinDesk. Available: https://www.coindesk.com/markets/2022/07/21/vitalik-buterin-discusses-ethereums-upcoming-merge-and-surge-at-ethcc-in-paris/ (accessed September 2023).

[LEG15] Le Goff, M., Carrier, C., & Walker, S. (2015). *Introducing stem: a new multichannel audio format*. In Proceedings of International Society for Music Information Retrieval Conference.

[LES00] Lessig, L. (2000). *Code is law*. Harvard magazine, 1, 2000.

[LI18] Li, R., Song, T., Mei, B., Li, H., Cheng, X., & Sun, L. (2018). *Blockchain for large-scale Internet of things data storage and protection*. IEEE Transactions on Services Computing, 12(5), 762-771.

[LI19] Li, C., & Palanisamy, B. (2019, June). *Incentivized blockchain-based social media platforms: A case study of steemit*. In Proceedings of the 10th ACM conference on web science (pp. 145-154).

[LI21] Li, R. (2021). *Fingerprint-related chaotic image encryption scheme based on blockchain framework*. Multimedia Tools and Applications, 80, 30583-30603.

[LIA17] Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., & Njilla, L. (2017). *Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability*. In 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (pp. 468-477). IEEE.

[LIA20] Liang, W., Fan, Y., Li, K. C., Zhang, D., & Gaudiot, J. L. (2020). *Secure data storage and recovery in industrial blockchain network environments*. IEEE Transactions on Industrial Informatics, 16(10), 6543-6552.

[LIG23] *A friendly Smart Contract Language for Tezos*. LigoLang. Available : https://ligolang.org/?lang=jsligo (accessed September 2023).

[LIU13] Liu, J., Huang, Z., Cai, H., Shen, H. T., Ngo, C. W., & Wang, W. (2013). *Near-duplicate video retrieval: Current research and future trends*. ACM Computing Surveys (CSUR), 45(4), 1-23.

[LIU21a] Liu, Z., Xiang, Y., Shi, J., Gao, P., Wang, H., Xiao, X., ... & Hu, Y. C. (2021). *Make web3.0 connected*. IEEE transactions on dependable and secure computing, 19(5), 2965-2981.

[LIU21b] Liu, L., Zhou, S., Huang, H., & Zheng, Z. (2021). *From technology to society: An overview of blockchain-based DAO*. IEEE Open Journal of the Computer Society, 2, 204-215.

[LOM22] Lombardi, R., de Villiers, C., Moscariello, N., & Pizzo, M. (2022). *The disruption of blockchain in auditing–a systematic literature review and an agenda for future research*. Accounting, Auditing & Accountability Journal, 35(7), 1534-1565.

[LOU23] *Le Louvre portal*. Le Louvre. Available : https://www.louvre.fr/en/online-tours (accessed September 2023).

[LU09] Lu, J. (2009). *Video fingerprinting for copy identification: from research to industry applications*. Media Forensics and Security, 7254, 725402.

[MA14] Ma, L., Montgomery, A. L., Singh, P. V., & Smith, M. D. (2014). *An empirical analysis of the impact of pre-release movie piracy on box office revenue*. Information Systems Research, 25(3), 590-603.

[MAD23] Madine, M., Salah, K., Jayaraman, R., & Zemerly, J. (2023). *NFTs for Open-Source and Commercial Software Licensing and Royalties*. IEEE Access, 11, 8734-8746.

[MAK23] *MakerDAO*. Available: https://makerdao.com/en/ (accessed September 2023).

[MAN22a] Manolache, M. A., Manolache, S., & Tapus, N. (2022). *Decision making using the blockchain proof of authority consensus*. Procedia Computer Science, 199, 580-588.

[MAN22b] Manoylov, MK. *OpenSea Reveals that over 80% of its free NFT mints were plagiarized, spam or fake.* (2022) The Block. Available: https://www.theblock.co/linked/132511/opensea-reveals-that-over80-of-its-free-nft-mints-were-plagiarized-spam-or-fake (accessed September 2023).

[MDN23] *JavaScript*. (2023) Mdn web docs. Available: https://developer.mozilla.org/fr/docs/Web/JavaScript (accessed September 2023).

[MEH19] Mehar, M. I., Shier, C. L., Giambattista, A., Gong, E., Fletcher, G., Sanayhie, R., ... & Laskowski, M. (2019). *Understanding a revolutionary and flawed grand experiment in blockchain: the DAO attack*. Journal of Cases on Information Technology (JCIT), 21(1), 19-32.

[MER23] *decentralization*. Merriam-Webster Available: https://www.merriam-webster.com/dictionary/decentralization(accessed September 2023).

[MIK17] Mik, E. (2017). *Smart contracts: terminology, technical limitations and real world complexity*. Law, innovation and technology, 9(2), 269-300.

[MIN19] Min, T., Wang, H., Guo, Y., & Cai, W. (2019, August). *Blockchain games: A survey*. In 2019 IEEE conference on games (CoG) (pp. 1-8). IEEE.

[MIN23] *Mintable Marketplace*. Available: https://mintable.app/ (accessed September 2023).

[MIR23] *Mirflickr25k*. Kaggle. Available : https://www.kaggle.com/datasets/paulrohan2020/mirflickr25k (accessed September 2023).

[MOL20] Mollah, M. B., Zhao, J., Niyato, D., Lam, K. Y., Zhang, X., Ghias, A. M., ... & Yang, L. (2020). *Blockchain for future smart grid: A comprehensive survey*. IEEE Internet of Things Journal, 8(1), 18-43.

[MON22] Monroe, R. *Coffeezilla, the YouTuber Exposing Crypto Scams*. (2022) The New Yorker. Available: https://www.newyorker.com/news/letter-from-the-southwest/coffeezilla-the-youtuber-exposing-crypto-scams (accessed September 2023).

[MOR22] Moreaux, A., & Mitrea, M. (2022). *Blockchain Assisted Near-duplicated Content Detection*. B2C Conference Proceedings, p.98, 2022

[MOR23a] Moreaux, A., & Mitrea, M. (2023). *Visual content verification in blockchain environments*. Blockchain and Cryptocurrency, Vol.1, Issue 1, pp. 44-55.

[MOR23b] Moreaux, A. C., & Mitrea, M. P. (2023). *Blockchain asset lifecycle management for visual content tracking*. IEEE Access.

[MOR23c] Moreaux, A. C., & Mitrea, M. P. (2023). *Royalty-friendly digital asset exchanges on blockchains*. IEEE Access vol. 11, pp. 56235- 56247.

[MPG19] *IoMT White Paper*. (2019) MPEG. Available : https://mpeg.chiariglione.org/standards/mpeg-iomt/iomt-white-paper (accessed September 2023).

[MUR23] Murray, A., Kim, D., & Combs, J. (2023). *The promise of a decentralized Internet: What is Web3 and how can firms prepare?*. Business Horizons, 66(2), 191-202.

[NAD05] Nadeem, A., & Javed, M. Y. (2005). *A performance comparison of data encryption algorithms*. In 2005 international Conference on information and communication technologies (pp. 84-89). IEEE.

[NAK04] *RPOW - Reusable Proofs of Work*. (2004) Satoshi Nakamoto Institute. Available: https://nakamotoinstitute.org/finney/rpow/ (accessed September 2023).

[NAK08] Nakamoto, S., & Bitcoin, A. (2008). *A peer-to-peer electronic cash system*. Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf, 4(2), 15.

[NAR16] Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton University Press.

[NFT23] *The NFT Bay*. Available: https://thenftbay.org/ (accessed September 2023).

[NOB99] Noble, J., Taivalsaari, A. K. P., & Moore, I. (1999). *Prototype-based programming: Concepts, languages and applications*. Springer.

[NST20] Rose, S., Borchert, O., Mitchell, S., Connelly, S. *Zero Trust Architecture*. (2020) NIST Special Publication 800-207. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf (accessed September 2023).

[NST23] NIST Portal. NIST. Available: https://www.nist.gov/ (accessed September 2023).

[NTS23] *NewtonScript Portal*. Available : https://newtonscript.org/ (accessed September 2023).

[OCD20] Series, O. B. P. (2020). *The tokenisation of assets and potential implications for financial markets*. The Secretary General of the OECD.

[OMI23] *Open Music Portal*. Available: https://open-music.org/ (accessed September 2023).

[OOS02] Oostveen, J., Kalker, T., & Haitsma, J. (2002). *Feature extraction and a database strategy for video fingerprinting*. In Recent Advances in Visual Information Systems: 5th International Conference, VISUAL 2002 Hsin Chu, Taiwan, March 11–13, 2002 Proceedings 5 (pp. 117-128). Springer Berlin Heidelberg.

[OPS23] *Opensea NFT Marketplace*. Available: https://opensea.io/ (accessed September 2023).

[OPZ23a] *OpenZeppelin Token Wizard*. Available: https://wizard.openzeppelin.com/ (accessed September 2023).

[OPZ23b] *ERC20*. OpenZeppelin docs. Available: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20(accessed September 2023).

[OST03] Ostrom, E. (2003). *Toward a behavioral theory linking trust, reciprocity, and reputation. Trust and reciprocity: Interdisciplinary lessons from experimental research*, 6, 19-79.

[OXF23] *Interoperability*. Oxford Learner's Dictionaries. Available: https://www.oxfordlearnersdictionaries.com/definition/english/interoperability (accessed September 2023).

[PET77] Peterson, J. L. (1977). *Petri nets*. ACM Computing Surveys (CSUR), 9(3), 223-252.

[POD01] Podilchuk, C. I., & Delp, E. J. (2001). *Digital watermarking: algorithms and applications*. IEEE signal processing Magazine, 18(4), 33-46.

[PWC21] *Study of the environmental impact of the Tezos blockchain Life Cycle Assessment of the Tezos blockchain protocol*. (2021) Nomadic Labs. Available: https://tezos.com/2021-12-06-Tezos-LCA-Final.pdf (accessed September 2023).

[QAD22] Qadir, S., Parker, G. *NFT Royalties: The $1.8bn Question*. (2022) Galaxy. Available: https://www.galaxy.com/research/insights/nft-royalties/ (accessed September 2023).

[QAS19] Qasse, I. A., Abu Talib, M., & Nasir, Q. (2019). *Inter blockchain communication: A survey*. In Proceedings of the ArabWIC 6th Annual International Conference Research Track (pp. 1-6).

[QUR20] Qureshi, A., & Megías Jiménez, D. (2020). *Blockchain-based multimedia content protection: Review and open challenges*. Applied Sciences, 11(1), 1.

[RAD18] Radomski, W., Cooke, A., Castonguay, P., Therien, J., Binet , E., Sandford, R. ERC-1155: Multi Token Standard. (2018) Ethereum Improvement Proposals. Available: https://eips.ethereum.org/EIPS/eip-1155. (accessed September 2023).

[RAM21] Rambhia, V., Mehta, V., Mehta, R., Shah, R., & Patel, D. (2021). *Intellectual Property Rights Management Using Blockchain*. In Information and Communication Technology for Competitive Strategies (ICTCS 2020) Intelligent Strategies for ICT (pp. 545-552). Springer Singapore.

[REM23] Remix Ethereum Portal. Available: https://remix.ethereum.org/ (accessed September 2023).

[REN21] Rendle A., McLean, C. *NFTs—A Question of Ownership*. (2021) Available: https://www.taylorwessing.com/en/interface/2021/copyright-update/nfts-a-question-of-ownership (accessed September 2023).

[RIN23] *Etherscan Portal*. Available: https://etherscan.io/tokens (accessed September 2023).

[ROD09] Rodriguez-Doncel, V., & Delgado, J. (2009). *A media value chain ontology for MPEG-21*. IEEE MultiMedia Magazine, 16(4), 44-51.

[ROD16] Rodríguez-Doncel, V., Delgado, J., Llorente, S., Rodríguez, E., & Boch, L. (2016). *Overview of the MPEG-21 media contract ontology*. Semantic Web, 7(3), 311-332.

[ROM14] Romanosky, S., Hoffman, D., & Acquisti, A. (2014). *Empirical analysis of data breach litigation*. Journal of Empirical Legal Studies, 11(1), 74-104.

[ROY23] Roy, S. S., Das, D., Bose, P., Kruegel, C., Vigna, G., & Nilizadeh, S. (2023). *Demystifying NFT Promotion and Phishing Scams*. arXiv preprint arXiv:2301.09806.

[RUM17] Rumalla, R. (2017). *UNPAID & MISSING ROYALTIES OF THE MUSIC INDUSTRY: FINDING THE BLACK BOX* (Doctoral dissertation, Berklee College of Music).

[SAL19] Salijeni, G., Samsonova-Taddei, A., & Turley, S. (2019). *Big Data and changes in audit technology: contemplating a research agenda*. Accounting and business research, 49(1), 95-119.

[SAN15] Santoso, F. K., & Vun, N. C. (2015). *Securing IoT for smart home system*. In 2015 international symposium on consumer electronics (ISCE) (pp. 1-2). IEEE.

[SAR09] Saracoglu, A., Esen, E., Ates, T. K., Acar, B. O., Zubari, U., Ozan, E. C., ... & Ciloglu, T. (2009). *Content based copy detection with coarse audio-visual fingerprints*. In 2009 Seventh International Workshop on Content-Based Multimedia Indexing (pp. 213-218). IEEE.

[SCH07] Schneier, B. (2007). *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons.

[SCH17] Scherer, M. (2017). *Performance and scalability of blockchain networks and smart contracts*.

[SCH23] Scharfman, J. (2023). *Decentralized finance (defi) fraud and hacks: Part 2*. In The Cryptocurrency and Digital Asset Fraud Casebook (pp. 97-110). Cham: Springer International Publishing.

[SGU21] Sguanci, C., Spatafora, R., & Vergani, A. M. (2021). *Layer 2 blockchain scaling: A survey*. arXiv preprint arXiv:2107.10881.

[SHA20] Sharma, P. K., Kumar, N., & Park, J. H. (2020). *Blockchain technology toward green IoT: Opportunities and challenges*. IEEE Network, 34(4), 263-269.

[SHI23] Shilina, S. (2023). *The future of social networking: Decentralization for user empowerment, privacy, and freedom from censorship*.

[SIN20] Singh, A., Parizi, R. M., Han, M., Dehghantanha, A., Karimipour, H., & Choo, K. K. R. (2020). *Public blockchains scalability: An examination of sharding and segregated witness*. Blockchain cybersecurity, trust and privacy, 203-232.

[SMI23a] Smith, C. *ERC-20 Token Standard*. (2023) Ethereum. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/ (accessed September 2023).

[SMI23b] Smith, C. *Blocks*. (2023) Ethereum. Available: https://ethereum.org/en/developers/docs/blocks/ (accessed September 2023).

[SOB12] Sobti, R., & Geetha, G. (2012). *Cryptographic hash functions: a review*. International Journal of Computer Science Issues (IJCSI), 9(2), 461.

[SOL21] Solaiman, E., Wike, T., & Sfyrakis, I. (2021). *Implementation and evaluation of smart contracts using a hybrid on-and off-blockchain architecture*. Concurrency and computation: practice and experience, 33(1), e5811.

[SOR23] Sor, J. *Who is Cofeezilla, the crypto detective who says he got Sam Bankman-Fried to admit to fraud*. (2023) Markets Insider. Available: https://markets.businessinsider.com/news/currencies/coffeezilla-sam-bankman-fried-ftx-bankruptcy-crypto-scam-detective-2023-1 (accessed September 2023).

[SPY23a] *SmartPy Portal*. Available: https://smartpy.io/ (accessed September 2023).

[SPY23b] *Meta-programming*. SmartPy. Available: https://legacy.smartpy.io/docs/introduction/meta_programming (accessed September 2023).

[SPY23c] Load balancing SmartPy Contract. SmartPy legacy. Available: https://legacy.smartpy.io/ide?cid=Qme4m6KAuLApTxuyJQvfi6NFTquKorDjkeipgJvAfUvYgk&k=94aa601729099a18a274 (accessed September 2023).

[STA23] *Media-Worldwide*. (2023) Statisa. Available : https://www.statista.com/outlook/amo/media/worldwide#revenue (accessed September 2023).

[SWI23] *SWIFT Portal*. Available: https://www.swift.com/ (accessed September 2023).

[SZA05] Szabo, N. (2005). *[*.

[SZA94] Szabo, N. (1994). *Smart contracts*.

[TAK18] Takenobu, T. (2018). *Ethereum EVM illustrated*. Github Pages.

[TAQ23] *Taquito TypeScript library*. Available: https://tezostaquito.io/ (accessed September 2023).

[TAT19] Tateishi, T., Yoshihama, S., Sato, N., & Saito, S. (2019). *Automatic smart contract generation using controlled natural language and template*. IBM Journal of Research and Development, 63(2/3), 6-1.

[TEM04] Temin, P. (2004). *Financial intermediation in the early Roman Empire*. The Journal of Economic History, 64(3), 705-733.

[TEZ23a] *FA1.2 asset smart contracts*. Tezos Gitlab. Available: https://tezos.gitlab.io/user/fa12.html (accessed September 2023).

[TEZ23b] *List of bakers*. Open Tezos. Available: https://opentezos.com/baking/bakers_list/ (accessed September 2023).

[TEZ23c] *Smart contract concepts*. Open Tezos. Available: https://opentezos.com/smart-contracts/smart-contracts-concepts/ (accessed September 2023).

[TEZ23d] *Liquid Proof-of-Stake*. Open Tezos. Available: https://opentezos.com/tezos-basics/liquid-proof-of-stake (accessed September 2023).

[THI22] Thibault, L. T., Sarry, T., & Hafid, A. S. (2022). *Blockchain scaling using rollups: A comprehensive survey*. IEEE Access.

[THO21] Thompson, C. (2021). *The untold story of the NFT boom*. The New York Times.

[THO22] Thorn, A., Marcantonio, M., Parker, G. *A Survey of NFT Licenses: Facts & Fictions.* (2022) Galaxy. Available: https://www.galaxy.com/insights/research/a-survey-of-nft-licenses-facts-and-fictions/ (accessed September 2023).

[TID21] Tidy, J. (2021) *Fake Banksy NFT sold through artist's website for £244k*. BBC. Available: https://www.bbc.com/news/technology-58399338 (accessed September 2023).

[TRE21] *Treasury Continues to Counter Ransomware as Part of Whole-of-Government Effort; Sanctions Ransomware Operators and Virtual Currency Exchange*. (2021) U.S. Department of the treasury. Available: https://home.treasury.gov/news/press-releases/jy0471 (accessed September 2023).

[TRE22] U.S. Treasury Sanctions Notorious Virtual Currency Mixer Tornado Cash. (2022) U.S. Department of the treasury. Available: https://home.treasury.gov/news/press-releases/jy0916 (accessed September 2023).

[TRU17] Trujillo, JL., Fromhart, S., Srinivas, V. *Evolution of blockchain technology Insights from the GitHub platform*. (2017) Deloitte Insights. Available: https://www2.deloitte.com/us/en/insights/industry/financial-services/evolution-of-blockchain-github-platform.html (accessed September 2023).

[TSE20] Tseng, L., Yao, X., Otoum, S., Aloqaily, M., & Jararweh, Y. (2020). *Blockchain-based database in an IoT environment: challenges, opportunities, and analysis*. Cluster Computing, 23, 2151-2165.

[TZP23] TZIP Gtilab. Available: https://gitlab.com/tezos/tzip (accessed September 2023).

[UND23] *Money Laundering*. United Nations Office on Drugs and Crime. Available: https://www.unodc.org/unodc/en/money-laundering/overview.html (accessed September 2023).

[VEN00] Venkatesan, R., Koon, S. M., Jakubowski, M. H., & Moulin, P. (2000). *Robust image hashing*. Proceedings 2000 International Conference on Image Processing. Vol. 3, pp. 664-666. IEEE.

[VER18] Brock, B. *How Do Ethereum Smart Contracts Work? It's Deceptively Simple*. (2018) Very. Available: https://www.verytechnology.com/iot-insights/how-do-ethereum-smart-contracts-work-its-deceptively-simple (accessed September 2023).

[VLE17] van Lee, H. S. (2017). *A Formalization of the Greater Fools Theory with Dynamic Epistemic Logic*. In Logic, Rationality, and Interaction: 6th International Workshop, LORI 2017, Sapporo, Japan, September 11-14, 2017, Proceedings 6 (pp. 585-597). Springer Berlin Heidelberg.

[VOG15] Vogelsteller, F., Buterin, V. *ERC-20 Token Standard*. (2015) Ethereum Improvement Proposals. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/ (accessed September 2023).

[WAC19] Wachter, S., & Mittelstadt, B. (2019). *A right to reasonable inferences: re-thinking data protection law in the age of big data and AI*. Colum. Bus. L. Rev., 494.

[WAN21a] Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). *Non-fungible token (NFT): Overview, evaluation, opportunities and challenges*. arXiv preprint arXiv:2105.07447.

[WAN21b] Wang, Y., Su, Z., Li, J., Zhang, N., Zhang, K., Choo, K. K. R., & Liu, Y. (2021). *Blockchain-based secure and cooperative private charging pile sharing services for vehicular networks*. IEEE Transactions on Vehicular Technology, 71(2), 1857-1874.

[WAN22] Wang, Y., Su, Z., Zhang, N., Xing, R., Liu, D., Luan, T. H., & Shen, X. (2022*). A survey on metaverse: Fundamentals, security, and privacy*. IEEE Communications Surveys & Tutorials.

[WAN23a] Wang, Y., Su, Z., & Yan, M. (2023). *Social Metaverse: Challenges and Solutions*. arXiv preprint arXiv:2301.10221.

[WAN23b] Wang, Y., Su, Z., Xu, Q., Li, R., Luan, T. H., & Wang, P. (2023). *A secure and intelligent data sharing scheme for UAV-assisted disaster rescue*. IEEE/ACM Transactions on Networking.

[WIL16] Wilson, C., & Yang, F. (2016). *Shanxi Piaohao and Shanghai Qianzhuang: a comparison of the two main banking systems of nineteenth-century China*. Business History, 58(3), 433-452.

[WIL17] Williams, R., McMahon, E., Samtani, S., Patton, M., & Chen, H. (2017). *Identifying vulnerabilities of consumer Internet of Things (IoT) devices: A scalable approach*. In 2017 IEEE International Conference on Intelligence and Security Informatics (ISI) (pp. 179-181). IEEE.

[WJS23] *Web3.js – Ethereum JavaScript API*. Available: https://web3js.readthedocs.io/en/v1.8.2/ (accessed September 2023).

[WOO14] Wood, G. (2014). *Ethereum: A secure decentralised generalised transaction ledger*. Ethereum project yellow paper, 151(2014), 1-32.

[WOO17] Woodside, J. M., Augustine Jr, F. K., & Giberson, W. (2017). *Blockchain technology adoption status and strategies*. Journal of International Technology and Information Management, 26(2), 65-93.

[WPY23] Web3py documentation. Available: https://web3py.readthedocs.io/en/stable/ (accessed September 2023).

[WU04] Wu, M., Trappe, W., Wang, Z. J., & Liu, K. R. (2004). *Collusion-resistant fingerprinting for multimedia*. IEEE Signal Processing Magazine, 21(2), 15-27.

[XIO22] Xiong, H.; Chen, M.; Wu, C.; Zhao, Y.; Yi, W. (2022) *Research on Progress of Blockchain Consensus Algorithm: A Review on Recent Progress of Blockchain Consensus Algorithms*. Future Internet , *14*, 47.

[XU22] Xu, X., Zou, G., Chen, L., & Zhou, T. (2022). *Metaverse space ecological scene design based on multimedia digital technology*. Mobile Information Systems, 2022.

[YAK18] Yakovenko, A. (2018). *Solana: A new architecture for a high performance blockchain v0. 8.13*. Whitepaper.

[YAN19] Yang, F., Zhou, W., Wu, Q., Long, R., Xiong, N. N., & Zhou, M. (2019). *Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism*. IEEE Access, 7, 118541-118555.

[YTB18] Cryptocurrencies: Last Week Tonight with John Oliver (HBO). (2018) YouTube, https://www.youtube.com/watch?v=g6iDZspbRMg (accessed September 2023).

[YTB22] *Right Clicking All The NFTs*. (2022) YouTube. Available: https://www.youtube.com/watch?v=i_VsgT5gfMc&t=11s (accessed September 2023).

[ZEN18] ZenCash Statement on Double Spend Attack. (2018) Horizen. Available: https://blog.horizen.io/zencash-statement-on-double-spend-attack/ (accessed September 2023).

[ZHA20a] Zhang, R., & Chan, W. K. V. (2020, July). *Evaluation of energy consumption in block-chains with proof of work and proof of stake*. In Journal of Physics: Conference Series (Vol. 1584, No. 1, p. 012023). IOP Publishing.

[ZHA20b] Zhang, Rong, and Wai Kin Victor Chan. (2020*) Evaluation of energy consumption in block-chains with proof of work and proof of stake*. Journal of Physics: Conference Series. Vol. 1584. No. 1. IOP Publishing.

[ZOU19] Zou, W., Lo, D., Kochhar, P. S., Le, X. B. D., Xia, X., Feng, Y., ... & Xu, B. (2019). *Smart contract development: Challenges and opportunities*. IEEE Transactions on Software Engineering, 47(10), 2084-2106.

[ZUP20] Zupan, N., Kasinathan, P., Cuellar, J., & Sauer, M. (2020). *Secure smart contract generation based on petri nets*. In Blockchain Technology for Industry 4.0: Secure, Decentralized, Distributed and Trusted Industry Environment (pp. 73-98). Singapore: Springer Singapore.

ECOLE DOCTORALE

Titre : Traçage du contenu visuel, droits de propriété intellectuelle & blockchain : de l'abstraction des processus à l'interopérabilité fonctionnelle

Mots clés : Blockchain, Contenu visuel, Droits de propriété intellectuelle, Interopérabilité, Confiance Numérique, Web3

Résumé : La thèse traite d'actifs (ou tokens) et de logiciels (ou Smart Contracts) présents sur la blockchain, à travers les domaines du contenu visuel et des droits à la propriété intellectuelle (DPI). Plus précisément, nous traitons d'abord les limites de distribution et de DPI des tokens puis au seuil d'exigences techniques et capacités de calculs limités des Smart Contracts avant d'aborder la centralisation de ces technologies autour de cadres applicatifs récurrents. Pour ce faire, nous avons développé des méthodologies palliant ces problèmes dont l'abstraction permet non seulement leur application à de nombreux domaines mais également leur interopérabilité mutuelle et avec des solutions conventionnelles. Ces outils et leurs associations permettent de répondre à des cas d'applications complexes, parmi lesquelles nous illustrons la protection de contenu visuel généré par des objets connectés ou la protection d'œuvres d'art mises à disposition par des musées dans des environnements web3.

Title : Visual content tracking, IPR management, & blockchain: from process abstraction to functional interoperability

Keywords : Blockchain, Visual Content, Intellectual Property Rights, Interoperability, Digital Trust, Web3

Abstract : The thesis deals with blockchain assets (or tokens) and software (or Smart Contracts), approached through the lens of visual content and Intellectual Property Rights (IPR). Specifically, we address the limited means of distribution of tokens, their weak IPR, the high knowledge requirement to create Smart Contracts, their limited computing capacities, and the centralization of both token and Smart Contract usage around specific environments and aplicative verticals. We thus develop a series of zero-trust abstract methodologies that tackle these limitations and that can be further implemented and interoperated with one another or with conventional solutions. These tools and their combinations can then answer complex use cases, the two of which we demonstrate are the protection of Internet of Media Things visual content or the end-to-end protection of museum-owned artwork in web3 environments.

Institut Polytechnique de Paris
91120 Palaiseau, France