Royalty-friendly digital asset exchanges on blockchains

Alexandre C Moreaux, Mihai P Mitrea

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.Doi Number

Royalty-friendly digital asset exchanges on blockchains

Alexandre C. Moreaux, and Mihai P. Mitrea
SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France

Corresponding author: Mihai Mitrea (e-mail: mihai.mitrea@telecom-sudparis.eu).

ABSTRACT The distribution of royalties associated with the exchange of digital assets, especially Non-Fungible Tokens (NFTs), is now more than ever a strong point of contention. Between conceptual disagreements and technical limitations, actors have implemented a variety of solutions tailored to their needs. In the process, creators and buyers have lost the possibility of transparent, trusted, and interoperable exchanges of said assets, often having to compromise to connect with the rest of the community. This study deals with the automatic distribution of royalty payments. We first investigate the current day limitations, and formally state their underlying requirements, before advancing a royalty-friendly NFT marketplace-agnostic trading framework. The advanced solution, referred to as the RM-TLSC – *Royalty Management Token-Level Smart Contract*, establishes synergies between the token and Smart Contract paradigms, thus ensuring royalties are managed throughout the life cycle of the asset. A comprehensive, open-source software implementation is provided for the Ethereum blockchain, while the generality of the approach is cross-checked by an open-source proof of concept for the Tezos blockchain. The effectiveness of the results is illustrated through a case-study related to ISO 21000-23 Smart Contracts for Media standard.

INDEX TERMS Backward compatibility, Blockchain, Ethereum, Interoperability, Intellectual Property Rights, Marketplace, Non-Fungible Tokens, Royalties, Smart Contract, Tezos.

I. INTRODUCTION

During the last three decades, increasingly more private and professional assets people own and manipulate have become partially or completely digital. Photos, videos, code, and online accounts, all of which are subject to intrinsic issues imposed by licences, royalties, or Intellectual Property Rights (IPR) management. The hyper distributed and decentralized paradigms such as web3.0 [1] and Distributed Ledger Technologies (DLTs, of which blockchains are examples of) have exacerbated these issues, creating a 10-figure market [2] of assets requiring protection. Specifically, digital art extensively suffers from the lack of transparency, reliability, and flexibility in IPR enforcement solutions.

When approaching the realm of IPR for digital art (be it audio, image, video, or any type of digital creative endeavor) a contradiction can be spotted. On the one hand, royalty laws and regulations come together with IPRs, both at the global and specific levels, as with the multimedia industry [3]. On the other hand, the business workflows and technical tools required to support these laws and regulations are met with suspicion and reluctance. For instance, the 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 somewhere between 250 million USD and 5 billion USD [4].

Blockchains, by their very nature, are an appealing solution for both ensuring trust and automating royalty management. Blockchains provide systematic and immutable transaction potential as well as unique custom-made exchangeable digital assets, referred to as *Non-Fungible Tokens* (NFTs). To bring visibility to their NFTs, creators usually rely on *marketplaces*, *i.e.*, decentralized applications that connect them to potential buyers. These marketplaces are the blockchain equivalent of stores and implement their own set of rules, fees, and royalty policies according to widely accepted standards that leave the management of royalties to the sellers. Ethereum marketplace royalties alone have already added up to an estimated value of 1.8 billion USD [5].

The fair and systematic distribution of royalties was a major argument in the forthcoming of NFTs since their inception [5]. Now that the global NFT market can be counted in tens of billions of USD [2], [5], [6], 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 main reason for this issue is that IPR laws need to be enacted to the world of blockchain which still suffers from regulatory grey points. Whilst most sellers do their best to follow ethical practices following their own criteria, others may have vested interests not to act in an ethical manner. Unfortunately, it is common to see NFTs involved in several types of fraud. The biggest NFT selling platform in the world, *OpenSea* [7], reported that over 80% of the assets created with their simplified "lazy minting" process were plagiarized works, fake collections, and spam [8]. The absence of a uniform and reliable *modus operandi* to rightfully compensate all parties prevents the establishment of a trusted global environment. Hence, specifying, designing, and programming an automatic, transparent, and trustworthy compensation ecosystem is one of the major challenges blockchain actors shall answer during the coming years. This challenge will require significant technical resources and answers to philosophical differences between actors, both of which require 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. Initiatives have emerged on large blockchains, *e.g.*, the Ethereum Improvement Proposal 2981 on the eponymous blockchain [9] but are still to hit the mainstream.

Beyond regulatory issues, this paper focusses on the technical issues related to automatic royalty distribution. It prospects current day approaches to royalty friendly NFT exchanges before advancing an interoperable solution whose applicative perimeter is discussed. It is structured as follows. Section II introduces the basic blockchain concepts subsequently used in the paper. Section III analyses current-day NFT exchange solutions, brings to light the multifold constraints related to NFT exchanges, and derives the underlying requirements for alleviating them. Section IV presents the advanced interoperable solution, at both conceptual and methodological levels. In Section V, the advanced solution is illustrated with a case study for the Ethereum blockchain. A proof-of-concept for the Tezos blockchain will serve as cross check for the generality of our approach. Section VI critically discusses our contributions and identifies their limitations, while Section VII concludes the paper and lays out future work.

While targeting an interoperable, flexible, and transparent solution for royalty-enabled digital asset exchanges on blockchains, the paper main contributions are:
(1) the identification of five requirements any such solution should meet (*cf*. Section III),
(2) The RM-TLSC (*Royalty Management Token-Level Smart Contract*) framework (*cf*. Section IV) that meets these requirements,
(3) a fully operational open-source implementation of the RM-TLSC for Ethereum (*cf*. Section V), and
(4) an open-source proof of concept implementation of the RM-TLSC for Tezos (*cf*. Section V). Note: the open-source implementations are made available upon paper publication.

II. BLOCKCHAIN IN A NUTSHELL

This section goes over fundamental blockchain concepts. It is by no means exhaustive and serves as a ground for the rest of the paper. The definitions in this section follow [10], [11], [12], [13], [14], [15], [16].

A. BLOCKCHAIN ARCHITECTURE

A blockchain is a network of nodes that generates cryptographically linked blocks of immutable data. These blocks are generated automatically at a fixed rate that depends on the specific blockchain, *e.g.*, approximately one per 10 minutes for Bitcoin [12], one per 15 seconds for Ethereum [13], or one per 30 seconds on Tezos [14]. Users of this network can transact with one another through anonymous accounts called *wallets*, signed into using unique private keys. When a transaction is called for by a user, it joins a pool of transactions that must be validated before their execution and inclusion in a block. The method of validation depends on the specific *consensus algorithm* of the blockchain. These algorithms elect users to take on the responsibility to validate transactions and reward them with compensation for their computational power, or *gas*, and fees. Some examples of consensus algorithms are *Proof of Work* (PoW) where candidates must solve a difficulty-adjusting cryptographic problem, *Proof of Stake* (PoS) which enables users to stake a portion of their assets in a frozen account to have a chance to validate a block, or *Proof of Authority* (PoA) which ties this stake to an individual's real-world identity. An analysis on the history and future of these consensus algorithms can be found in [16], while [17] and [18] compare their uses and performances. In short, the historic consensus method is PoW but its inefficiency, poor scalability, and environmental negative consequences prompted the shift to PoS. This architecture, coupled with the anonymity of wallets, provides an environment where parties can securely exchange assets without needing to trust each other *i.e.*, a *zero-trust* environment. The security of blockchains come from the combination of the cryptographic encryption, the consensus algorithms, and the redundance of data on every node.

B. APPLICATIVE LAYER

Although Bitcoin was built to support its cryptocurrency, most blockchains are now transversal and built to support an array of applicative use cases (e.g., [12], [13]). The foundational aspect of which is the *Smart Contract*, a piece of code which lives on the blockchain and runs exactly as it is programmed, with no possibility of change. Smart Contracts are developed in different languages on different blockchains, *e.g.*, Solidity on Ethereum or Michelson in Tezos (Michelson being conceptually low level, developers typically use higher level languages such as SmartPy [19]). These Smart Contracts can be combined with frontend User Interfaces (UI) to create *Decentralized Applications* (dApps). dApps differ from legacy apps in that they have a trustless, verifiable, and tamper-proved behavior, complete data integrity, perfect privacy, and do not have any downtime. In addition, although each blockchain has its own logic, Smart Contracts tend to be open, transparent, and accessible through an address, just like wallets. This way, users are allowed to build off each other's work, similarly to Application Programming Interfaces (APIs) in off chain environments. On the downside, Smart Contracts cannot be maintained, scale very poorly, can cause and suffer from network congestion in addition to not always being user friendly. This overall behavior leads users to rely on third party software they do not verify. A technical state-of-the-art of the Smart Contract paradigm can be found in [20] and [21].

C. TOKENS

A *token* is the blockchain representation of a digital asset. All transactions on blockchains use tokens in a form or another. On one hand, *transactional tokens* (*e.g.*, ETH for the Ethereum blockchain or tez for the Tezos blockchain) correspond to the local currency of the environment. They are used to compensate miners who contribute their computing power and to pay for whatever one wishes. On the other hand, blockchain users can also create tokens to represent digital assets such as a piece of art, a diploma, a proof of ownership of a physical asset, or even a ticket to an event. These creations are typically divided into two categories: *Fungible Tokens* (FTs) and *Non-Fungible Tokens* (NFTs), although hybrid standards are becoming increasingly popular. FTs can be split and are interchangeable (*e.g.*, a specific transactional token is worth as much as any other, just like a dollar) whereas NFTs are unique, cannot be split and are notably used as representations of digital art and sold as such. Regardless of their type, tokens are standardized to include basic functionalities on top of which users can create additional ones. Tokens can be traded from a wallet to another for transactional tokens in a *sale* or moved from a wallet to another belonging to the same person in a *transfer*. Inter-blockchain communication is a highly discussed topic [22], and cross-blockchains transfers, although not widespread at the time of writing, have started emerging [23].

D. INTEROPERABILITY EFFORTS

Although international standard bodies (ISO, IEEE) and blockchain-specific organizations (INATBA) have published sets of recommendations for blockchains [24], [10], [25], the most important guidelines to developers and users are by far application-level standards such as [26], [27]. These standards are blockchain specific and are typically built by the community itself. Their goal is to set trusted precedents for new initiatives to build on. Examples include best practices, libraries, or so-called standard token contracts that act as secure and open ledgers for digital assets to be accounted for. Standard token contracts enable the proper functioning of tokens, not only tracking suppliers and holders but also allowing for secure transfers. For instance, Ethereum relies on ERC – *Ethereum Request for Comments* standards [26] that anyone can create, granted they explain their idea clearly and foster enough community support. These applicative standards differ from EIP – *Ethereum Improvement Proposals* [27] that aim core functions of Ethereum. For instance, the ERC20 standard defines the functions and events related to fungible tokens [14], as illustrated in Figure 1. These standards are then implemented [28] and widely used. The Tezos blockchain uses a similar system called the TZIP – Tezos Improvement Process [29] that notably gave birth to the FA – *Financial Application* [30] token standards.

FIGURE 1. ERC20 standard methods and events [14].

E. SUMMARY

This section dealt with fundamental blockchain concepts from the basic architecture to token standards. It serves to illustrate the existence of widely used working standards. In practice, the applicative spectrum of blockchains is not only extremely vast, but also relates to various applicative verticals. For instance, blockchain can enhance UAVs in disaster rescue [31], Smart Grid data sharing [32], Electric Vehicles [33], Donating [34], Multimedia Content Protection [35], amongst others. The present paper focuses on royalty-enabled blockchain asset exchanges, of which we investigate the current-day solutions in the next Section.

III. CURRENT DAY DIGITAL ASSET EXCHANGES

Through an in-depth analysis of royalty management on blockchain marketplaces, this section identifies design requirements for the methodological workflow presented in Section IV. Thus, we investigate five complementary aspects: (a) the relationship between the notions of tokens and ownership, (b) the diversity and relevance of marketplaces, (c) the differences in their respective approaches to royalty distribution, (d) the efforts put forth to bring interoperability, and (e) the current limitations of existing solutions. This section is informed by [5], [7], [9], [36], and a private communication with J. Seibel [37], co-author of the EIP2981 proposal.

A. OWNERSHIP

The notion of token ownership is complex, even misunderstood by dedicated collectors. In fact, NFT art collectors do not directly purchase the art displayed on marketplaces but a combination of a digital token containing metadata, that often points to off chain storages where the artwork is stored, and a license that establishes exploitation rules. Almost no NFT provides IP ownership and a sizable portion only allow personal use or creative commons licenses, which gives non-buyers as many rights as buyers [36]. These issues as well as the necessary changes that could enable a healthier environment for IP transfers are extensively discussed in [36]. A natural consequence of this fact is that royalty rights are even more contentious. Whilst some marketplaces and actors are trying to enable a royalty-friendly environment, others are attempting to dissociate the ideas of royalties and NFTs [5]. 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 yet, nor will it be within our study.

Given the ambiguity of the notion of ownership in the blockchain ecosystem, the *1st Requirement we set for our solution is that it shall allow users to clarify their respective positions on IPR and royalties before the first transaction.*

B. MARKETPLACES

NFT marketplaces are some of the most popular forms dApps take. They are digital platforms for buying, selling, and sometimes even minting (creating) NFTs. While it is true that users could mint their own NFT and directly send them to other address, a vast majority of the trading volume happens via marketplaces. This is the case not only because these platforms bring user-friendliness but also because they allow users to advertise their creations, a particularly arduous task for an individual with no prior following in an environment designed for anonymity. Each marketplace implements its own set of rules, regulations, fees, and royalty policies. Some marketplaces allow for direct purchases for fixed prices (in cryptocurrency and/or legal tender), while others operate open auctions, or focus on NFT-for-NFT trading [38].

In short, marketplaces are varied, and their motives and means of operation solely depend on those who created them. The only common aspect to all marketplaces is their *modus operandi*, which typically resembles some variation of a *Swap Contract*, which functions as illustrated in Figure 2. First, a Smart Contract is initialized by the platform with the desired set of rules. It contains a mapping of tokens to their respective owners and prices, illustrating in Figure 2'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 2. A buyer who wants to purchase a token on the marketplace can then use the Smart Contract's *buy()* function [14], [15], [30], Step 3 Figure 2. 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 2) before sending the desired token to the buyer (Step 3.D. Figure 2). These operations happen under the hood of user-friendly interfaces developed by the marketplace. An example of a full implementation can be found in [39].

The above explanation is simplified and shows the central aspect of the platform. Notably, tokens typically stay on a given marketplace, which allows for secondary sale traffic to also occur on that marketplace. This secondary market is far from negligeable. 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 [5].

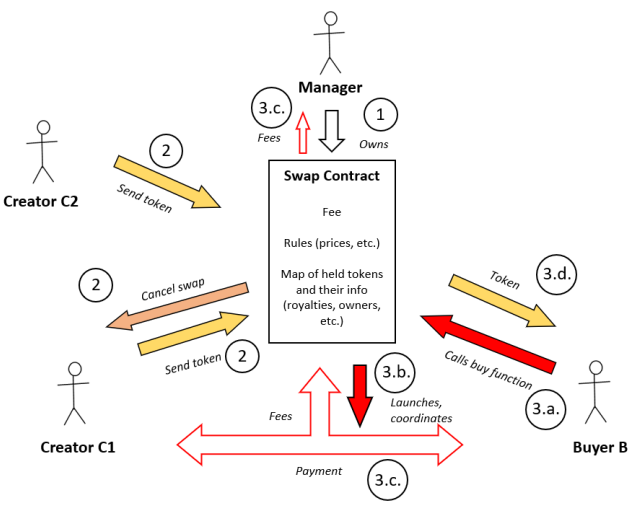


FIGURE 2. Simplified Swap Contract workflow. Numbers represent steps, in order; yellow arrows show token movement; red arrows show purchase execution; white and red arrows show royalty distribution.

The marketplace ecosystem is hands off for creators who only need to send their token to the Smart Contract while specifying its price. This advantage is traded for the trust the user must have in the platform, or the resources to verify it.

Moreover, the creator must compensate the marketplace *via* a fee, is bound by all the rules and limitations the marketplace wishes to impose, and no longer has a say on the life of the token after the initial transaction.

Considering the technical and applicative heterogeneity on current day marketplaces, the *2nd Requirement we set is that our solution shall be marketplace agnostic.*

C. ROYALTY DISTRIBUTION APPROACHES

Currently, two approaches to distributing royalties with token sales exist: *within* the asset or *alongside* it. The former handles royalties as part of its standard *e.g.*, its *sell()* function automatically redirects part of the sale to the original minter. The later uses non-royalty-enabled tokens, simply managing the royalty aspect separately *e.g.*, *via* a Smart Contract. Having a separate entity handling royalties could sound like a lackluster solution compared to the immutability of having royalty information in the token itself. Yet, it allows for flexibility for buyers and sellers that is simply not possible with unmodifiable hardcoded constraints. Given that no widespread royalty-enabled standards exist at the time of writing, marketplaces are left to implement royalties by themselves. 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.

Some marketplaces do in fact compensate artists for every transfer of their NFT if it occurs on the platform. For instance, *OpenSea* does allow secondary market royalties that are paid by the seller, not the buyer. Of course, platforms have no power to enforce their rules once the token has left their borders.

For fair compensation to cross the borders of marketplaces, royalty information must appear in the token itself. But if it appears, why not simply enforce sub-payments in the very *transfer*() function of the token? First, the solution would not suit every user and user-case. Moreover, the biggest problem automatic royalty payments must face is that it is impossible to differentiate sales from transfers. Users can and, in fact, do move assets, which turns systematic enforcement into an extremely limited solution. Unfortunately, any exceptions allowing such transfers would immediately be exploited by malicious actors looking to bypass fair compensation.

Hence, a deep protocol, engrained in the logic of the blockchain itself is required to bypass this issue. This potential solution would have to 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 misidentify a sale for a transfer? This unfortunate situation pushes intra-token enforcement out of the picture, leaving initiatives such as Ethereum Improvement Proposal (EIP) 2981 [9], focused on a non-enforced standard, at the forefront of this discussion. We would also like to mention that royalty distribution in a software licensing NFT use cases has been explored in [40].

Under these circumstances, the *3rd Requirement is that our approach shall ensure royalties are paid out indeterminately while ensuring cordial retraction if the parties come to an agreement.*

D. ETHEREUM IMPROVEMENT PROPOSAL (EIP) 2981

Ethereum Improvement Proposal (EIP) 2981 [9] 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 3. According to this solution, the burden of this extra cost should lie on the consumer, and that the value of assets should be contingent on royalties. Metadata information is added after the creation of the token, as shown in Step 1 Figure 3. After that, the token can be used freely, including being sent to a marketplace Swap Contract (Step 2 Figure 3, as in Figure 2). Upon purchase, the marketplace can enforce the payment detailed in the metadata if they wish to (Step 3 Figure 3).

Although users advanced unique features and extensions [41], the EIP2981 authors decided to keep nothing but the lowest common denominator in order 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 the acceptance of standard. The simple EIP needed a 12-month validation process, a particularly 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 marketplaces will eventually be pressured by artists and the public into adopting the standard, a point from which legal arguments for royalty distribution could be in the realm of possibilities. The *Mintable* [42] and *Coinbase* [43] platforms have adopted EIP2981 while *OpenSea* have signaled they will be implementing it soon. Some upstart exchanges have seized the opportunity to get some market shares by ignoring royalty practices and displaying lower prices but the biggest traders with known addresses cannot use them whilst maintaining their reputation. As of now, marketplaces diversely handle royalty payouts, some preferring real time payments while others staggering them to save on gas.

We set a *4th Requirement for our solution that shall enforce systematic royalty payments rather than leaving this enforcement to a third party.*

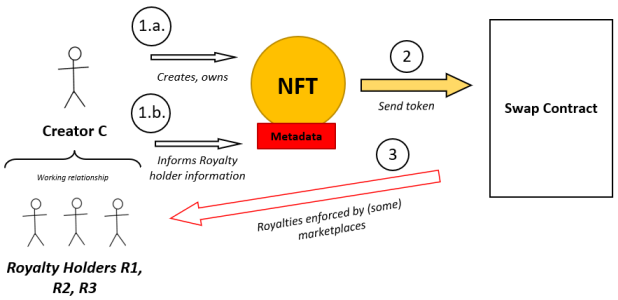


FIGURE 3. EIP2981 workflow. Numbers represent steps, in order; yellow arrows show token movement; red arrows show purchase execution; white and red arrows show royalty distribution.

E. BARRIERS TO ADOPTION

Before leaning into policy adoptions, it is important to identify the reasons why some actors are still reluctant about the blockchain space. The main barriers towards industrial blockchain adoption are identified by [44], 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 where 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 paper, it is particularly interesting to note that *intellectual property concerns* ranked as the most popular third reason. The issue, although occasionally clouded by the more endemic issues associated with blockchains, is at the back of many minds. Newer studies [45], [46] report interoperability concerns to be the biggest limit to blockchain adoption.

When it comes to barriers to adoption of royalty-friendly systems, two points can be highlighted. 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 [47]. This climate does not provide fertile grounds for building solid and widely accepted regulations. The second is the latency brought by the process itself. Solutions need to be thought out, discussed, developed, adjusted before they even need to go through the standardization process, implementation, and acceptance.

Given these limitations, we set as a *5th Requirement that our solution shall be backward compatible towards already existing standards.*

F. SUMMARY

A retrospective look at Section III shows that an interoperable, fair, transparent, and scalable way to exchange assets would alleviate most concerns new and to-be blockchain adopters have. Whilst fully interoperable blockchains are an objective to aim for, the current state-of-the-art exposes major obstacles in this respect. Section IV proposes an intermediate step by advancing a fully interoperable intra-blockchain solution, allowing for the fair exchange of assets within an ecosystem. The solution will meet the five above-identified requirements, namely:
Requirement 1: it shall allow users to come to agreements having clarified their respective positions on IPR and royalties before the first transaction.
Requirement 2: it shall be marketplace-agnostic.
Requirement 3: it shall ensure royalties are paid out indeterminately while ensuring cordial retraction if the parties come to an agreement.
Requirement 4: it shall enforce systematic royalty payments rather than leaving this enforcement to a third party.
Requirement 5: it shall be backwards compatible toward already existing standards.

IV. METHODOLOGICAL WORKFLOW

A. METHOD SYNOPSIS

As explained in Section III.C, royalty distribution solutions are divided into two categories: token-centric and Smart Contract-centric. At their core, they must enable multiple *sub-transactions* (royalty payments) to occur upon an NFT purchase while balancing reliability, transparency, and efficiency. Potential solutions also need to take past and future into account, ensuring backward compatibility and the flexibility to ensure interoperability with future developments. Our solution respects this design philosophy by referencing the five requirements listed in Section III.

On the one hand, token standard-based solutions are limited in their flexibility and solving their main pitfall undermines decentralization. On the other hand, Smart Contracts, although being the *de facto* solution, cannot follow the full life cycle of tokens. To circumvent both issues, we specified and designed a synergetic approach, further referred to as the *RM-TLSC – Royalty Management Token-Level Smart Contract*.

As opposed to managing the exchange of tokens at the level of the seller, the RM-TLSC is initialized with a set of rules and is exchanged alongside the token as to follow it during its entire life cycle, handling royalties as intended by the creator, as illustrated in Figure 4. This way, *Requirements 1*, *2*, and *3* are met. Note that the RM-TLSC also includes retraction rights management as to provide a proper lifecycle to the Smart Contract, as dictated by *Requirement 3*. Specifically, we decided to use RM-TLSC as a capsule, owning the token and ensuring every transaction follows the rules that it has been initialized with to satisfy *Requirement 4*. In alignment with *Requirement 5*, the heart of RM-TLSC uses the functionalities brought by current day working standards.

To ensure enforceability (*Requirement 4*), we must prevent the bypassing of rules by malicious users simply calling the standard *transfer* method. Given that modifying this cornerstone function would render the token standard

noncompliant (which would contradict *Requirement 5*), we elected to modify the owner instead. In essence, the RM-TLSC acts as a "mini swap contract" that belongs to the current holder of the token, acting as a zero-trust third party.

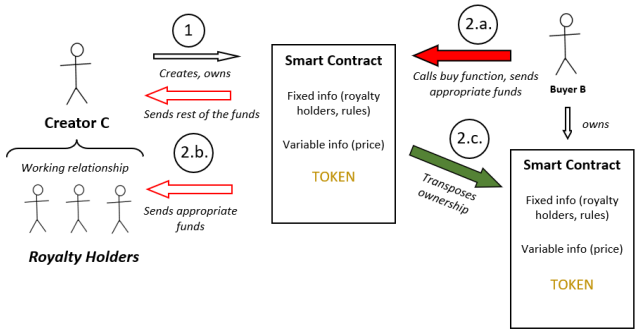


FIGURE 4. Advanced solution: the *RM-TLSC – Royalty Management Token-Level Smart Contract*. Numbers represent steps, in order; white arrows show ownership; red arrows show purchase execution; white and red arrows show royalty distribution; green arrows show changing of hands of the TLSC.

B. DETAILED DESCRIPTION

In practice, this solution is completely flexible, and users can not only customize but also add features or rules to their instance (*Requirement 1*). The lifecycle of RM-TLSC encompasses three steps:

- *Initialization*: The purpose of this phase is to create the RM-TLSC with the wanted set of rules.
- *Trading*: In this phase, the RM-TLSC is traded normally.
- *Termination*: This phase allows for a backdoor in case the RM-TLSC is no longer an appropriate solution.

The *Initialization phase* consists in instantiating the RM-TLSC with the set of rules to be applied (*e.g.*, "5% of each transaction of this token goes to this wallet address") before sending the token to it. To minimize potential mistakes, the RM-TLSC will only accept the specific token it was built for, and only after its address is explicitly approved by the token owner. This approval is necessary as the RM-TLSC is not allowed to request the token beforehand, as is defined by token standards [14][15] (*Requirement 5*). Before the first sale, we remain in the initialization phase, hence all the rules are still modifiable by the owner. In this exposition of the framework, any token can use this wrapper, but real-life implementations could apply restrictions, *e.g.*, only allowing the original minter of a token to send it to this RM-TLSC. This sequence of events if illustrated in Figure 5.

The RM-TLSC's concept uses tools available on most application-enabled blockchains, and the specificity of the implementation limits itself to the programming language and application-level standards of the environment. Section V presents detailed implementations for the Ethereum and Tezos blockchains.

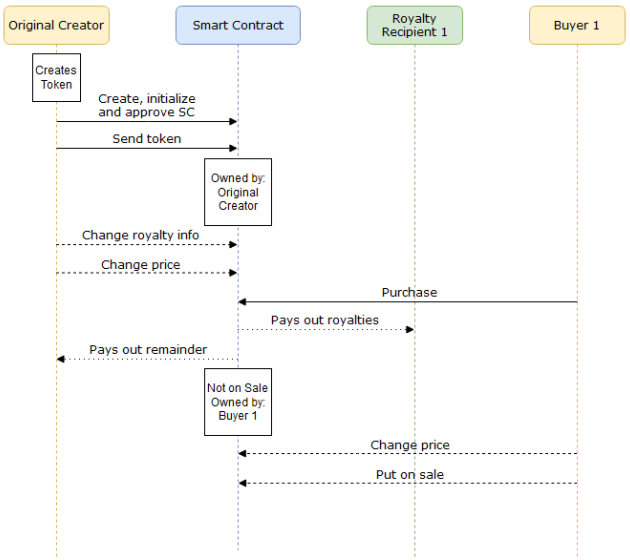


FIGURE 5. Sequence diagram for the *Initialization* phase of the RM-TLSC, from instantiation to first purchase.

The *Trading phase* begins after the first transfer, from which point onwards the price is the only modifiable royalty-related field. In our illustrative implementation (Section V), this constraint applies even if the token is sent back to the original owner. This means that a token can be flipped for many times and the prices can vary, but the distribution of royalties and limitations are locked (*Requirement 3*). This philosophy resembles EIP2981 [9], [41], 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 simply transact the currently listed price to the Smart Contract with the buy function (*Requirement 2*). 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 implementation in Section V pays out the royalties in real time but staggered or on-demand payments are also possible. The buyer becomes the new owner and can change the price if they wish to. The Smart Contract acts as the NFT itself, changing hands or resting in someone's possession. Figure 6 shows the n^{th} purchase of the token.

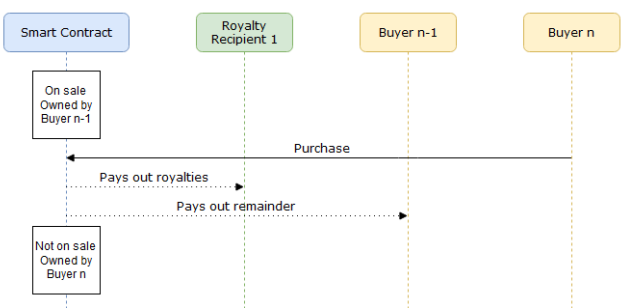


FIGURE 6. Sequence diagram for the trading phase of the RM-TLSC, showing automatic royalty payment.

Finally, if a current owner wishes to extract the token from its capsule, the retraction rules set by the original 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 their acceptance through a transaction to the *retraction* function. The willingness of royalty holders to let go of their guaranteed income and subsequent negotiations are left to the concerned parties and is not handled by RM-TLSC. Once this function successfully called by the current owner, 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 7. The specifics of the retraction phase are set during the *Initialization phase*.

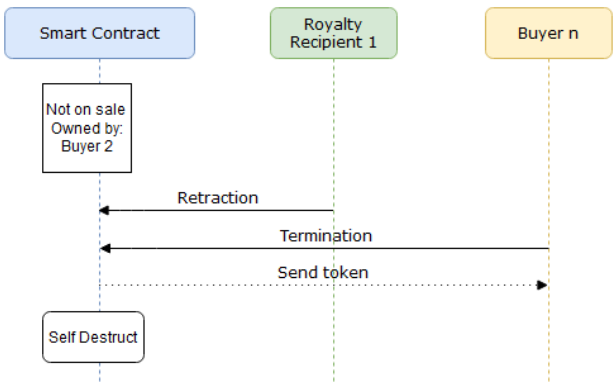


FIGURE 7. Sequence diagram for the termination phase and end of life cycle of the RM-TLSC. Hereafter, the token bears no trace of the proposed solution.

C. SUMMARY

This Section established a flexible, transparent, and interoperable framework that meets the requirements set forth in Section III.

The solution is both marketplace and blockchain agnostic. Indeed, we rely on the fundamentals of blockchains, and do not require specific tools in implementing and deploying a RM-TLSC. Moreover, this solution is highly customizable to any given use case, particularly when it comes to the conditions of the *Initialization phase* and triggering the *Termination phase*.

Table I provides a point-by-point comparison of the RM-TLSC's characteristics with regards to standard marketplace Swap Contracts and EIP2981-enabled exchanges. The five first comparison points address the requirements summarized in Subsection III.F, while the two last deal with supplementary aspects (namely the existence of fees and the zero-trust nature). Although we did not set requirements linked to these features, the RM-TLSC bears advantages in their regard, as we explained throughout this Section. A deeper analysis of the advantages and limitations of the RM-TLSC can be found in Section VI.

TABLE I
COMPARATIVE SUMMARY OF THE RM-TLSC AGAINST STATE-OF-THE-ART SOLUTIONS ON KEY POINTS DETAILED IN SECTION III.

	State-of-the-art solutions		**Advanced solution**
	Swap Contract (Figure 2)	**EIP2981 (Figure 3)**	**RM-TLSC (Figure 4)**
Flexible royalty structure	No	No	Yes
Marketplace agnostic	No	Yes	Yes
Indeterminate support	No	Yes	Yes
Royalty enforcement	Some	No	Yes
Standard tokens	Yes	Yes	Yes
Charges fees	Yes	No	No
Zero-trust	No	Not enforced (informative only)	Yes

The conceptual pillars of RM-TLSC (namely, the Token and the Smart Contract) are widely standardized, accepted, and used. Given that our architecture does not bring extra constraints to these concepts, it can be thought of in terms of backward and forward compatibility. In the next Section, we illustrate this flexibility by going into detail with an ISO 21000-23 inspired example implementation.

V. CASE STUDY

A. SCENARIO OVERVIEW

In this Section, we discuss an illustrative implementation of the RM-TLSC. It is structured as a walkthrough of the implementations and their usage.

The scenarios we use to run the tests were inspired by the ISO 21000-23 Smart Contracts for Media standard. This standard relates to standardized Smart Contracts produced from standard RDF ontologies and XML schemas, used to codify IPR information related to music and visual information [48].

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, *creator* is not entitled to subsequent sales portions and will receive compensation only for the initial sale. This

is summarized in Table II. The 80% remaining share will go to the seller in context.

TABLE II
SUMMARY OF THE PARTIES USED IN THE SCENARIOS OF SECTION V.

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%

We start with a complete Ethereum implementation, selected thanks to the activity of the community and the royalty-related discussions it boasts. We then go through a proof-of-concept implementation on the Tezos blockchain in order to show that the workflow we bring forth in Section IV is highly adaptable not only to use cases but also to blockchains.

The specific use case illustrates the ins and outs of this approach. We will use the Ethereum implementation to discuss the specifics of the RM-TLSC and the Tezos implementation to explore a specific sequence of events that could occur with the RM-TLSC, although both are available for the two blockchains. 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.

B. RM-TLSC ETHEREUM IMPLEMENTATION

In this example, we use the popular ERC721 NFT standard [15], although any other would function similarly. The complete Smart Contracts are available alongside this paper, including a complete Truffle framework alongside a test NFT and web3py [49] 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 lifecycle of the token successfully, the other highlighting fraudulent behavior not being tolerated by the RM-TLSC. The experiments ran in this Subsection used the settings summarized in Table III.

TABLE III
ENVIRONMENT, STANDARDS, AND VERSIONS USED IN THE ETHEREUM IMPLEMENTATION OF THE RM-TLSC.

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

Let us now dive into specifics. The Smart Contract stores three kinds of variables: the ones relevant to the owner 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, its 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 owner, price, and retraction check which are dynamic. These are illustrated in the context of the scenario explained hereafter in Figure 8.

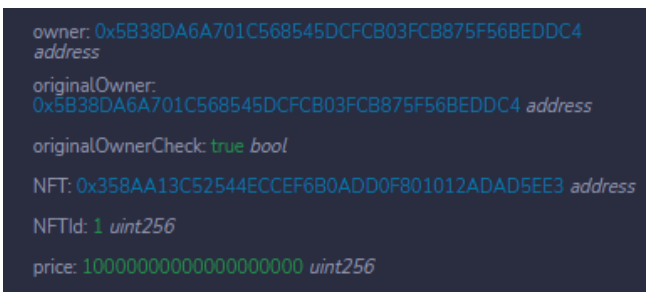


FIGURE 8. Owner and token-related variable states after an Ethereum test scenario initialization.

Two modifiers, namely *onSale* and *onlyBy* will ensure functions are only called by authorized addresses when the owner wishes to. They are shown in Figure 9.

```
modifier onlyBy(address authorizedAddress){
    require(msg.sender == authorizedAddress ,"Sender not authorized.");
    _;
}

modifier onSale(){
    require(onSaleBool == true ,"Token not for sale");
    _;
}
```

FIGURE 9. Modifiers in RM-TLSC 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 to create the Smart Contract. Interestingly, it includes the specific token down to the Id as 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 [50] interface.

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

- *setup*: allows the owner of the token to initialize the Smart Contract and store their token. It requires the caller to be the owner of the token 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 and only if all rights holders have previously forfeited their rights by using the retraction function from their respective addresses. Once called, it begins the *Termination* phase.

For our tests, we used the simplest integration of the ERC721 standard. This sort of token can easily be created with a tool such as the OpenZeppelin online wizard [51]. We deployed these Smart Contracts on a 3-node, Hyperledger Besu EEA (Enterprise Ethereum Alliance [52])-compliant PoA private blockchain deployed on an Amazon Web Services sever, as well as on the now deprecated Rinkeby Ethereum testnet accessed through the Infura node cluster. We interacted with the Smart Contracts using the web3py library. Before manipulating the Smart Contract, we import various wallets to act as the different parties we will need (*creator*, *IP holders*, *buyers*, *etc.*). We then mint an ERC721 token which we setup a RM-TLSC for before having it go through the *Trading* and *Termination* phases. We illustrate the behavior using a test scenario on the Remix IDE [53]. This online IDE allows to write and tests Smart Contracts without requiring lengthy setups. We provide two scenarios in the form of 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 sequence of actions:

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 basic usage of the Smart Contract is illustrated trough an example transaction displayed in Figure 10.


```
(7) {
        "value": "0",
        "inputs": "(uint256)",
        "parameters": [
                "2000000000000000000"
        ],
        "name": "changePrice",
        "type": "function",
        "to": "0xf8e81D47203A594245E36C48e151709F0C19fBe8",
        "abi": "0xd33d0ea713df5cf21aa0a82ab8ddcebe3fa0a142aeed3af695acdebbce82099e",
        "from": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB"
}
```


FIGURE 10. The receipt of an Ethereum transaction to the *changePrice()* function.

The second, more complete *scenario.json* goes through the same steps, while additionally trying unauthorized actions in between. Namely, attempted purchases for the wrong price, attempts to change royalty information, and unsuccessful terminations. Figure 11 illustrates the first of these failures.

```
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Wrong price sent".
Debug the transaction to get more information.. Execution failed at 6
```

FIGURE 11. A failed Ethereum transaction due to the wrong price being sent.

C. RM-TLSC TEZOS PROOF OF CONCEPT

To illustrate the flexibility of this solution, we setup a proof-of concept implementation on a less populated infrastructure: Tezos [54]. Tezos is a developing yet established infrastructure with a lively update cycle, active marketplaces with different stances on royalty management, and low gas costs. We use SmartPy [19], an online IDE for Tezos Smart Contracts available through a Python library. This code is accessible alongside the Ethereum implementation. The experiments ran in this Subsection used the settings shown in Table IV.

TABLE IV
ENVIRONMENT, STANDARDS, AND VERSIONS USED IN THE TEZOS IMPLEMENTATION OF THE RM-TLSC.

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

A test scenario illustrating a basic utilization of the RM-TLSC, including attempted unauthorized actions, is available to run via SmartPy's "Run Code" button. This scenario illustrates the *Initialization*, *Trading*, and *Termination* phases, and is shown in Figure 12.

```
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=c1creator)
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 12. A SmartPy test scenario for a Tezos *RM-TLSC* implementation.

As discussed in the Ethereum example, the function testing in these scenarios is not exhaustive, but shows the basic functionalities and logic of the Smart Contract. Voluntary failures illustrate the behavior that is not accepted 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 13.

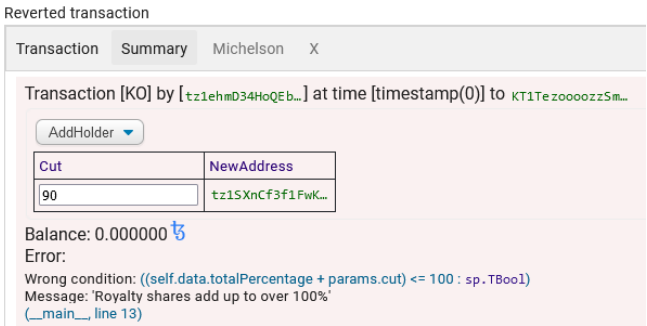


FIGURE 13. A failed attempt at adding royalty information from a Tezos user.

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 14, confirm that 20% of the price were indeed sent to *IPholder1*'s address and that the owner's address changes appropriately to *buyer1*'s.

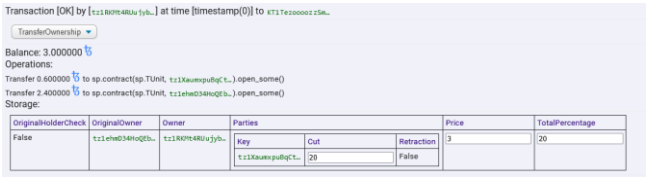


FIGURE 14. A successful purchase transaction showing successful royalty distribution on Tezos.

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 15, the current owner *buyer2* can terminate the TLSC.

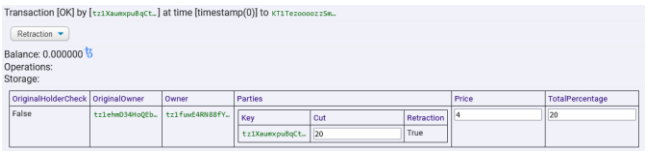


FIGURE 15. The successful retraction of the royalty holder on a Tezos RM-TLSC.

D. CONCLUSION

This section demonstrated the flexibility of the RM-TLSC with respect to blockchains and its independence to marketplaces: note that the case study has been successfully performed without any reference to a specific marketplace while blockchain specificities are limited to their programming language. The tools used in this paper can be found across a vast majority of application-driven blockchains, enabling this solution to be quickly implemented across varied environments. The software structure presented here is also modular, enabling users build upon base functionalities to tailor the solution to their specific needs.

VI. METHOD ANALYSIS

The RM-TLSC, introduced in Section IV and illustrated through open-source software tools in Section V, was conceived to answer the five requirements listed at the end of Section III. Consequently, the RM-TLSC features a *backwards-compatible*, *marketplace-agnostic framework* that ensures royalties are paid out *indeterminately* and *systematically*, according to *rules* that are *transparently set* beforehand and allowing *cordial retraction*.

While providing these beneficial features and being by design open to further extensions, the RM-TLSC still has limitations and shortcomings, some of which are inevitable

in the blockchain environment. In fact, a lot 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 *Notion of ownership*. The RM-TLSC framework relates to the notion of owning a token, given that rather than owning the token directly, users own a Smart Contract that in its 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 the field of digital art [55].

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.* When it comes to the question of whether digital assets should be subject to royalties, we will not go further than the value of assets being contingent of 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 not of major concern while in others it defeats the very purpose of an automatized framework. The RM-TLSC does not fall victim to it, hence compromising by enforcing royalties systematically. We then enabled the cordial retraction feature to add flexibility with regards to the systematic nature of the RM-TLSC. In particular, the examples brought forward in Section V enable owners to set the RM-TLSC's price to zero, thus allowing them to move assets between their wallets. This system could be abused but can be prevented with a minimum price clause.

⑤ *Technical knowledge 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 not only to automatize token minting, but also providing graphically intuitive solutions for the public to create tokens themselves [7], [51].

⑥ *Integration with existing solutions.* From the general point of view of NFT integration, the RM-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 is one that 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 only accepting the creator of the underlying token as an author. Tokens themselves have included standards to avoid similar behavior [14], [15], [30].

⑧ *Other TLSC use cases.* The notion of a rule-enforcing, zero-trust, third party can be used for many other applications than royalty management. In fact, the TLSC paradigm can apply to any use case requiring complete lifecycle rule enforcement. For instance, some media content could be restricted in access with regards to addresses or time [56]. 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 strongly benefit from the controlled access of restricted content enabled by an TLSC. A detailed analysis on the foundational notions of the Metaverse can be found in [57], while challenges of the social aspect of metaverse can be found in [58]. The TLSC's automatic rule enforcement can also benefit the myriad of applicative verticals mentioned in Subsection II.E. (namely [31], [32], [33], [34], and [35]), amongst others.

⑨ *Security/zero trust*. The RM-TLSCs 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 flexibility of the solution, further use and standardization efforts could patch any newly discovered flaws. RM-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 RM-TLSC by removing the need of a trusted third party and by requiring the explicit approval of the user 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 a 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 a safe one due to its blockchain nature, granted users remain 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. Whilst 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.

VII. CONCLUSIONS AND PERSPECTIVES

Platform-dependency and lack of interoperability are major yet overshadowed blockchain pain points. Although initiatives such as EIP2981 are gaining traction, their influence on overall token traffic is yet to be demonstrated. This paper's key contributions consist of (1) the identification of key requirements towards interoperable royalty friendly asset distribution, (2) the design and specification of the Royalty Management Token Level Smart Contract paradigm, and (3) the open-source RM-TLSC implementations for Ethereum and Tezos. The RM-TLSC is a flexible and interoperable framework that can follow tokens during their entire life cycle. We also discuss the applicative perimeter of this solution, displaying its capabilities, limitations, and relationship with other technical solutions in and out of the blockchain landscape.

Although the ideas advanced in this paper cannot completely solve the looming issues of proper compensation on blockchains, they contribute a unique approach, inviting new work and eventual robust standards to emerge. Specifically, further efforts will investigate enabling this solution to handle multiple tokens, making it gas effective, and expanding it into a fully interoperable model for inter-blockchain exchanges.

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 a properly 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.

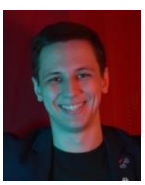
ACKNOWLEDGMENT

We acknowledge Najah Naffah for his insights on the world of applicative blockchains. We acknowledge James Seibel for the insightful discussion about royalty-enabled tokens.

REFERENCES

[1] S. Yang, M. Li, "Web3.0 Data Infrastructure: Challenges and Opportunities," in *IEEE Network*, vol. 37, no. 1, pp. 4-5, January/February 2023, doi: 10.1109/MNET.2023.10110018.
[2] "Blockchain Market Size […]." Fortune Business Insights. https://www.fortunebusinessinsights.com/industry-reports/blockchain-market-100072 (accessed Sep. 2022).
[3] "Royalty Law: Everything You Need to Know." Upcounsel. https://www.upcounsel.com/royalty-law (accessed Jan. 2023).
[4] E. Christman. "Just How Much Money Is There in Unclaimed Black Box Royalties?" Billboard. https://www.billboard.com/pro/unclaimed-black-box-royalties-how-much-money/ (accessed Jan. 2023).
[5] S. Qadir, G. Parker. "NFT Royalties: The $1.8bn Question." Galaxy. https://www.galaxy.com/research/insights/nft-royalties/ (accessed Dec. 2022).
[6] "Global Non-Fungible Token (NFT) Market Size to Reach USD 20 billion by 2028 | BlueWeave Consulting." GlobeNewswire. 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 Nov. 2022).
[7] Opensea NFT marketplace. https://opensea.io/ (accessed Aug. 2022).
[8] @opensea,Twitter. https://twitter.com/opensea/status/1486843201352716289?s=20&t=00yC9u6K4ktQ9PCNfwuV3A (accessed Sep. 2022).
[9] Z. Burks, J. Morgan, B. Malone, J. Seibel, "EIP-2981: NFT Royalty Standard," Ethereum Improvement Proposals, no. 2981, September 2020. [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-2981.
[10] "IEEE Blockchains Standards." IEEE Xplore. https://innovate.ieee.org/ieee-blockchain-standards-collection/ (accessed Nov. 2022).
[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system", 2008, [online] Available: https://bitcoin.org/bitcoin.pdf.
[12] V. Buterin, 2014. "A next-generation smart contract and decentralized application platform",White Paper.
[13] L. M. Goodman, 2014, "Tezos — a self-amending crypto-ledge", White Paper.
[14] "ERC-20 Token Standard." Ethereum https://ethereum.org/en/developers/docs/standards/tokens/erc-20/ (accessed Aug., 2022).
[15] W. Entriken, D. Shirley, J. Evans, N. Sachs, "EIP-721: Non-Fungible Token Standard." Ethereum Improvement Proposals, no. 721, January 2018. [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-721
[16] K. Azbeg, O. Ouchetto, S. Jai Andaloussi, L. Fetjah, "An Overview of Blockchain Consensus Algorithms: Comparison, Challenges and Future Directions", 2021, In: Saeed, F., Al-Hadhrami, T., Mohammed, F., Mohammed, E. (eds) Advances on Smart and Soft Computing. Advances in Intelligent Systems and Computing, vol 1188. Springer, Singapore. https://doi.org/10.1007/978-981-15-6048-4_31
[17] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," 2018 *41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, 2018, pp. 1545-1550, doi: 10.23919/MIPRO.2018.8400278.
[18] S. M. H. Bamakan, A. Motavali, A. B. Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria", Expert Systems with Applications, Volume 154,2020,113385,ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2020.113385.
[19] SmartPy Tezos IDE. https://smartpy.io/ (accessed Sep. 2022).
[20] Z. Zheng, S. Xie, H-N.Dai, W. Chen, X. Chen, J. Weng, M. Imran, "An overview on smart contracts: Challenges, advances and

platforms", Future Generation Computer Systems, Volume 105,2020, Pages 475-491,ISSN 0167-739X, https://doi.org/10.1016/j.future.2019.12.019.
[21] V. Y. Kemmoe, W. Stone, J. Kim, D. Kim, and J. Son, "Recent Advances in Smart Contracts: A Technical Overview and State of the Art", IEEE Access, vol. 8, pp. 117 782-117 801, 2020.
[22] "What is IBC (Inter-Blockchain Communication Protocol)?" Cosmos Developer Portal. https://tutorials.cosmos.network/academy/3-ibc/1-what-is-ibc.html (accessed May 2023)
[23] M. Borkowski, M. Sigwart, P. Frauenthaler, T. Hukkinen, S. Schulte, "Dextt: Deterministic Cross-Blockchain Token Transfers," in IEEE Access, vol. 7, pp. 111030-111042, 2019, doi: 10.1109/ACCESS.2019.2934707.
[24] "ISO/TC 307 Blockchain and distributed ledger technologies." ISO. https://www.iso.org/committee/6266604.html (accessed Nov. 2022).
[25] International Association for Trusted Blockchain Applications. https://inatba.org/ (accessed Feb. 2023).
[26] "ERC." Ethereum Improvement Proposals. https://eips.ethereum.org/erc (accessed Jan. 2023).
[27] "EIPs." Ethereum Improvement Proposals. https://eips.ethereum.org/ (accessed Aug. 2022).
[28] "ERC20." OpenZeppelin. https://docs.openzeppelin.com/contracts/4.x/erc20 (accessed May 2023)
[29] "Tezos Improvement Process (TZIP)." tz Wiki. https://wiki.tezos.com/learn/governance/tezos-improvement-process-tzip (accessed Jan. 2023).
[30] "FA2 A unified token contract interface." Tezos. https://tezos.b9lab.com/fa2 (accessed Feb. 2023).
[31] Y. Wang, Z. Su, Q. Xu, et al., "A Secure and Intelligent Data Sharing Scheme for UAV-Assisted Disaster Rescue.", 2022, arXiv:2211.12988.
[32] Y. Wang et al., "SPDS: A Secure and Auditable Private Data Sharing Scheme for Smart Grid Based on Blockchain," in IEEE Transactions on Industrial Informatics, vol. 17, no. 11, pp. 7688-7699, Nov. 2021, doi: 10.1109/TII.2020.3040171.
[33] Y. Wang et al., "Blockchain-Based Secure and Cooperative Private Charging Pile Sharing Services for Vehicular Networks," in IEEE Transactions on Vehicular Technology, vol. 71, no. 2, pp. 1857-1874, Feb. 2022, doi: 10.1109/TVT.2021.3131744.
[34] E. -C. Chang and N. -C. Tai, "Non-Fungible Token Donation Platform Delivers Continuous Goodwill without Daunting Donor Commitment", 2022 IEEE 11th Global Conference on Consumer Electronics (GCCE), Osaka, Japan, 2022, pp. 852-853, doi: 10.1109/GCCE56475.2022.10014392.
[35] A. Qureshi, D. M. Jiménez, "Blockchain-Based Multimedia Content Protection: Review and Open Challenges", 2021, Applied Sciences 11, no. 1: 1. https://doi.org/10.3390/app11010001
[36] A. Thorn, M. Marcantonio, G. Parker. "A Survey of NFT Licenses: Facts & Fictions." Galaxy. https://www.galaxy.com/research/insights/a-survey-of-nft-licenses-facts-and-fictions/ (accessed Jan. 2023).
[37] J. Seibel, private communication, Aug. 2022.
[38] "NFT marketplace list." NFT Innovation. https://www.nft-innovation.com/marketplace-list (accessed Feb. 2023).
[39] Steamr-dev/marketplace-contracts. GitHub. https://github.com/streamr-dev/marketplace-contracts/
[40] M. Madine, K. Salah, R. Jayaraman, J. Zemerly, "NFTs for Open-Source and Commercial Software Licensing and Royalties," in *IEEE Access*, vol. 11, pp. 8734-8746, 2023, doi: 10.1109/ACCESS.2023.3239403.
[41] "Discussion for EIP-2981: NFT Royalty Standard." GitHub. https://github.com/ethereum/EIPs/issues/2907 (accessed May 2023)
[42] Mintable marketplace. https://mintable.app/ (accessed May 2023)
[43] Coinbase marketplace. https://nft.coinbase.com/ (accessed May 2023)
[44] S. Davis, H. Arslanian, D. Fong, A. Watkins, W. Gee, C.Y. Cheung "PwC's Global Blockchain Survey 2018." https://theblockchaintest.com/uploads/resources/PwC%20-%20Global%20Blockchain%20Survey%202018%20-%202018.pdf (accessed Aug. 2022).
[45] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A Survey on Blockchain Interoperability: Past, Present, and Future Trends", 2021, ACM Comput. Surv. 54, 8, Article 168 (November 2022), 41 pages. https://doi.org/10.1145/3471140
[46] D. Appelbaum, E. Cohen, E. Kinory, S. S. Smith, "Impediments to Blockchain Adoption", Journal of Emerging Technologies in Accounting 1 September 2022; 19 (2): 199–210. https://doi.org/10.2308/JETA-19-05-14-26
[47] N. De., "Crypto-Mixing Service Tornado Cash Blacklisted by US Treasury." https://www.coindesk.com/policy/2022/08/08/crypto-mixing-service-tornado-cash-blacklisted-by-us-treasury/ (accessed Dec. 2022).
[48] "ISO/IEC 21000-23:2022 Information technology — Multimedia framework (MPEG-21) — Part 23: Smart Contracts for Media." ISO. https://www.iso.org/standard/82527.html (accessed Nov. 2022).
[49] Web3py documentation, Web3Py. https://web3py.readthedocs.io/en/v5/ (accessed Oct. 2022).
[50] Binodnp/openzeppelin-solidity. GitHub. https://github.com/binodnp/openzeppelin-solidity/blob/master/docs/IERC721Receiver.md (accessed Oct. 2022).
[51] OpenZeppelin token Wizard. https://wizard.openzeppelin.com/ (accessed Jan. 2023).
[52] Enterprise Ethereum Alliance. https://entethalliance.org/ (accessed Sep. 2022).
[53] Remix Ethereum IDE. https://remix.ethereum.org/ (accessed Jan. 2023).
[54] Tezos. https://tezos.com/ (accessed Dec. 2022).
[55] A. Rendle, C. McLean, "NFTs – a question of ownership." TaylorWessing. https://www.taylorwessing.com/en/interface/2021/copyright-update/nfts-a-question-of-ownership (accessed Feb. 2023).
[56] 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, https://doi.org/10.1016/j.icte.2021.08.009
[57] Y. Wang *et al*., "A Survey on Metaverse: Fundamentals, Security, and Privacy," in *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 319-352, Firstquarter 2023, doi: 10.1109/COMST.2022.3202047.
[58] Y. Wang, Z. Su, M. Yan, "Social Metaverse: Challenges and Solutions.", 2021, ArXiv, abs/2301.10221, https://doi.org/10.48550/arXiv.2301.10221.



Alexandre C. Moreaux is a PhD student at Institut Polytechnique de Paris. He holds an MS degree in digital imaging from Telecom SudParis and worked as a guest researcher in NIST's network department. He is an active contributor to ISO/IEC 21000-23 Smart Contracts for Media and ISO 23093 family - Internet of Media Things.

Mihai P. Mitrea holds an HDR (Habilitation à diriger des recherches) degree Pierre and Marie Curie University in Paris (2010) and a PhD from Politechnica University in Bucharest (2003). He is currently Associate Professor at Telecom SudParis. His research interest area cover to multimedia content tracking (watermarking, fingerprinting and blockchain). He is vice-president of the Cap Digital's Technical Commission on Digital Content and serves as advisor for the French delegation at ISO/IEC JTC1 SC29 (a.k.a. MPEG). Inside MPAI standardization organization, he is coordinating Neural Network Watermarking efforts.