A Tale on Decentralizing an App: the Case of Copyright Management

Andrea Vitaletti[1], Marco Zecchini[2,*]

[1]Sapienza Università di Roma, Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Rome, Italy
[2]University of Salerno, Dipartimento di Ingegneria dell'Informazione ed Elettronica e Matematica Applicata, Fisciano, Italy

Abstract

The Interested Party Information (IPI) system uniquely identifies the rights holders worldwide, making it possible to know for each subject and at any time which rights are protected, by whom and for which territories. Currently, this service is provided in a centralized way but, in 2021, Italian Society of Authors and Editors (SIAE) deployed a blockchain-based solution to completely decentralize this database to provide greater guarantees to the rights holders as well as to end users. With the development of the blockchain technology hosting this solution, Algorand, we can design techniques that reduce some trust assumptions of the solution developed by SIAE, enhancing at the same time its efficiency. In this paper, we show decentralized protocols to issue new on-chain rights representation consistently with the ledger's previous blocks, in particular, avoiding or discouraging to represent twice the same right. This makes rights holders autonomous in creating on-chain rights representations.

Keywords

Consistency Check, Blockchain Identity, Copyright Management, Decentralized Applications

1. Introduction

Creative operas express an intellectual work to which the law, like any other job, ensures compensation, protecting the birth and life of such works. Copyright is the mean by which creative works are protected and valued and its management is fundamental for the proper compensation of the people creating operas, i.e., rights holders.

Centralized Copyright Management and the roles of Collective Management Organizations. Managing copyright and related rights individually may not always be realistic. For instance, an author, performer, or producer cannot contact every radio station to negotiate licenses and remuneration for the use of their songs. On the other side, it is not practical for a radio station to seek specific permission from every rights holder for the use of each song. A Collective Management Organization[1] (CMO) facilitate rights clearance in the interest of both parties and economic reward for rights holders. By authorizing or mandating professional

DLT 2023: 5th Distributed Ledger Technology Workshop, May 25-26, 2023, Bologna, Italy
*Corresponding author.
vitaletti@diag.uniroma1.it (A. Vitaletti); mzecchini@unisa.it (M. Zecchini)
0000-0003-1074-5068 (A. Vitaletti); 0000-0002-2280-9543 (M. Zecchini)

[1]https://www.wipo.int/copyright/en/management/.

CMOs to manage their rights, rights holders can simplify the management of those rights. The main goals of a CMO are monitoring when, where, and what works are used and collecting the fees from users and distributing these to the rights holders.

This has led to a centralized governance of copyright management and CMOs are the epicenter. However, such a highly centralized architecture requires trust from end users. Indeed, rights holders fully delegate CMOs without having the opportunity to verify the correctness of their behavior. Furthermore, only a limited set of operators are involved in the royalties management sector. This centralized governance allows CMOs and a few other stakeholders to have the exclusive availability of valuable information that generates relevant revenues creating an advantageous position for business investments with respect to the other stakeholders.

While it is absolutely clear the advantage of delegating to trusted third parties all the cumbersome physical activities related to the negotiation and the collection of fees, the delegation should be transparent to provide the highest guarantee to all the involved parties and foster a competitive market. When transparency is a key requirement, there is no doubt that public/permissionless blockchains provide a clear advantage.

An opportunity of decentralization. In the last years, we saw the rise of decentralized applications. In these applications, unlike traditional applications, a set of peers agree on a common system state through an algorithmic consensus instead of relying on trusted third parties, thus providing unprecedented guarantees of transparency. The adoption of this kind of application in copyright management can reduce the influence of CMOs acting as intermediaries between end users.

Blockchain technologies are the enablers of such decentralized applications. Adopting blockchain for royalties management is considered one of the most appealing use cases envisioned for this technology[2]. We can imagine a future where while listening to a song on an online player, a transaction on the blockchain is automatically triggered to remunerate almost in real-time the rights holders in a suitable cryptocurrency. Thus, blockchain can replace CMO's job in the collection of revenues [1]. In addition, blockchain technology is completely transparent by design, creating a more dynamic and fair environment and providing new opportunities for all copyright-related stakeholders.

While, in principle, we can envision the employment of decentralized technologies in all the copyright management value-chain, a more realistic approach suggests we focus on key components of such value-chain to prove the viability of the proposed approach. For this reason, in this work, we will focus on the IPI database, which is in charge of recording the CMOs delegated by right holders to handle some rights in specific territories (see Section 2 for further details)

Identity Management and the need for a Trusted Party. On a blockchain, every user is typically identified by a public cryptographic key univocally associated with a secret key. The pair of cryptographic keys is created locally and randomly by each user, that, therefore, can create as many on-chain identities as he/she wants (i.e., pseudo-anonymity). Although verifying the correctness of a cryptographic signature guarantees the on-chain user identity

[2]https://consensys.net/blockchain-use-cases/

(i.e., the user knows the secret key associated with the public key), it does not link an on-chain identity to an entity in the physical world.

This problem affects many real use-case scenarios, including the one described in this paper. Indeed, a rights holder can create many on-chain identities, but there is no guarantee that these are associated with his/her real-world identity. For this reason, we need trusted entities, like CMOs, certifying the association between the real-world identity and the digital one. Moreover, in expectation of a fully digital economy based on tokens where all payments are transparent and automatized, revenue collection in the physical world also requires trusted entities in charge of this task. Observe that this introduces again the need of trusted third parties. However, we do believe that in this scenario is acceptable to rely on a centralized entity to guarantee the identity of a user, provided that the user is fully autonomous in managing his/her assets in a fully decentralized way.

Note that CMOs can be natural certification authorities to guarantee the identity of the right holders.

Towards a consistent on-chain rights management. In March 2021, the Italian Society of Authors and Editors (SIAE), the Italian CMO, issued around 4 million digital tokens representing the rights of the rights holders in one or more countries [2]. The purpose of SIAE was to decentralize the ipi system[3]. The IPI system uniquely identifies the rights holders worldwide, making it possible to know for each subject and at any time which rights are protected, by whom and for which territories.

The goal of this prototype was to show on a blockchain how to represent the rights in one or more countries and how rights holders can claim which CMOs is protecting their interest. The rights are represented on-chain through Non-Fungible Token (NFT) and those who possess on-chain the tokens represent the CMO in charge of protecting the interest related to the specific rights. In a centralized solution, the trusted party in charge of maintaining the database creates the right only if it does not lead the system to an inconsistent state (e.g., the right already exists). In a decentralized platform, we need a strategy to interpret two or more on-chain representations of the same right or deny their creation by design. From a general perspective, a key question in this context is how to check that the information we wish to insert on-chain is *consistent* with the previous blocks of the ledger. Note that this problem is related to the *double-spending* one.

The first approach, developed by SIAE, accepts in the blockchain all the transactions, but honest users, through a shared off-chain decoding technique (i.e., only the last record on-chain is valid), discard those transactions that do not respect the common rules. However, such an approach leads to a significant computational effort because it requires to always scan the entire blockchain to reach the final and valid status of the system. Another approach adopts a set of entities, defined as *issuers*, creating the tokens on the rights holder's behalf, but this requires trust assumptions similar to those of a centralized solution.

Smart contracts come to aid because they allow verifiable decentralized computations on a blockchain. As a first approach, they can be developed to discourage bad behaviors of the issuers through a penalization mechanism. Before issuing new tokens, an issuer transfers a

[3]https://www.ipisystem.org/

cryptocurrency deposit to the smart contract as a guarantee of its good behavior. Indeed, if caught in error (e.g., issuing twice the same right), he/she will lose the deposit. Other blockchain users look for this type of inconsistency because they gain the issuer's deposit if they detect one. However, with such an approach, we do not deny by design the creation of illicit rights. Indeed, smart contracts can be devised to verify on-chain by themselves if a right can be created. This can be done by maintaining specific data structures (e.g., bit matrix) to keep track of created rights in the blockchain. Unfortunately, smart contracts cannot address a large memory space and cannot handle complex computations. [3] introduces the notion of *computational effectiveness*, measuring the capacity of smart contracts to execute complex computations at reasonable costs. In particular, SIAE's tokens were issued on Algorand blockchain [4] whose smart contract cannot compute complex tasks due to both limitations in the size of a transaction and in the complexity of computations. In addition, Algorand smart contracts are more limited than other blockchain technologies (e.g., Ethereum [5]) [3]. For these reasons, we want to explore a solution where smart contracts verify the validity of proofs computed by users off-chain, proving their legitimacy in creating a new right on-chain (e.g., if every right is represented as a leaf in a Merkle Tree, the user would submit a Merkle trie proving that a specific leaf has an empty value).

Contribution of the paper. In this paper, we first analyze the current implementation of a decentralized version of the Interested Party Information (IPI) database. This solution has been implemented in Algorand and has been significantly influenced by the continuous and rapid evolution of such technology. It is sufficient to mention that at the beginning of the project, Algorand only provided stateless smart contracts (i.e., the current smart signatures). The current state of such technology allows us to envision interesting developments and to generalize the problem. In view of such development, the solution proposed in the paper evolves towards the design of a decentralized solution to make rights holders autonomous in issuing their rights while still maintaining the system in a valid state and reducing or avoiding double-spending.

We have designed three protocols: a first approach optimistically accepts all created rights as valid but allows users to destroy and penalize malicious users with economic penalizations eventually; a second approach organizes the smart contract storage space as a matrix of created rights to deny by design the creation of already created tokens; the last approach stores in the smart contract a succinct representation (i.e., Merkle Tree) of the created rights. With this last approach and a simple workaround on identity management (see Section 3.4 for more details), we allow users to manage the entire set of creatable rights of the IPI database autonomously and to do that consistently with already created ones.

Structure of the paper

The paper is structured as follows. In Section 2, we describe the solution developed by SIAE in 2021 and which are the trust assumptions for this approach. Motivated by such assumptions, in Section 3, we describe the decentralized solutions to perform consistency checks on new on-chain rights representation. Section 4 compares this work with related ones. Section 5 concludes the paper.

Heading	Short description	Cardinality	Example
$base\_no$	Interested Party (IP) base number, unique identifier of a right holder		I-001068130-6 identifying the rights holder
$name\_no$	IP name number, additional identifier of a rights holder		00334284961 identifying a pseudonym of the rights holder
$name$	IP name		Rossi Mario
cmo	Collecting Society	∼200	SIAE, SUISA
cc	Creation class	16	Musical Work, Dramatic work
ro	Role	34	Musical creator, Book publisher
ri	Right	26	Performing right, Re-transmission right,
$valid\_from$	The date from which the right management is given to the collecting society		01.01.2017
$valid\_to$	The date until which the right management is given to the collecting society		31.12.2017
$share$	The percentage of right the CMO is in charged to collect		100%, 75%, ...
$terr$	The territory in which the CMO is in charge of the right management	220	Italy, France, Europe, North America, World

Table 1
IPI main fields.

2. Modelling the IPI system

The Interested Party Information (IPI) System is an international information system containing data about all rights holders used by CMOs[4]. The IPI system's primary goal is to globally identify a natural person or legal entity acting in various creation classes with some roles and protected in some rights. This is important not only for the exchange of data between CMOs but also for the worldwide exchange of information with third parties and user organizations such as radio and TV stations.

The most relevant information in an IPI record is summarised in Table 1.

In the current centralized IPI System, the tuple $\langle cc, ro, ri\rangle$ identifies an *artistic right* of the rights holder. Specifically, the tuple $\langle cc, ro, ri\rangle$ identifies a right (ri) relative to a category of artistic works (cc) in which the rights holder has collaborated with a role (ro). The tuple $\langle name\_no, cmo, cc, ro, ri, share,\ terr, valid\_from, valid\_to\rangle$ represents an *agreement* between the rights holder ($name\_no$) and a copyright collecting society (cmo) valid in the interval of time $[valid\_from, valid\_to]$, and for the territory $terr$. Essentially, these fields mean that, in the interval of time from $valid\_from$ to $valid\_to$ and for the territory $terr$, the $name\_no$ has assigned to the collecting society cmo a percentage $share$ of the management of his/her artistic right $\langle cc, ro, ri\rangle$.

A rights holder is associated with only one IP base number and with one or more IP name number values (each IP Name has its own IP name number value).

Some fields of an IPI record are categorical and finite (such as cc, ro, ri, cmo, $terr$). Table 1 indicates the cardinality of those fields.

[4]Those that belong to International Confederation of Societies of Authors and Composers (CISAC)

2.1. Current blockchain-based IPI system solution

In March 2021, SIAE deployed a first solution to represent the IPI system on a blockchain [2]. The goal of this solution was to represent on-chain rights and agreements according to the definition described in this section. We will refer to the approach used by SIAE as the reference solution.

This solution represents an artistic right for a specific territory (i.e., the tuple $\langle cc, ro, ri, terr \rangle$) as a single *digital token*. Since rights are not tradable objects, these tokens must be represented as Non-Fungible Token (NFT). Each right must be protected by a CMO, formalized in an agreement between the parties. To implement the stipulation of an agreement, a rights holder transfers the property of the NFT to the target CMO. Therefore, the entity that possesses on-chain the NFT is in charge of protecting the specific user right in the real world. The user, however, should always be able to take the NFT back from the CMO in some specific scenario (for instance, to make an agreement with another CMO) while, on the other hand, a CMO should give its consent in the agreement.

The reference solution requires that the adopted blockchain supports the possibility of managing NFT for representing an artistic right for a specific territory. Since all the operation of creations (and, eventually, update/delete) must be agreed upon by multiple parties, namely the rights holder and a CMO, it requires that the adopted blockchain supports *multisignature accounts*. These accounts need the cryptographic signature of multiple standalone accounts to publish new transactions on the blockchain.

SIAE deployed its prototype on the Algorand mainnet [2]. Algorand was chosen as blockchain technology for deployment because it offers the above-mentioned features and it is currently considered one of the top performers in terms of scalability. Its inter-block time is very short: about 5 seconds and its consensus algorithm can confirm up to 1000 transactions per second [6]. This aspect will ensure good system performance if the network load increases if many rights holders adopt it or other functionalities are developed on top of IPI system.

In Algorand, the digital tokens are called Algorand Standard Asset (ASA). ASAs are composed of immutable fields (such as Creator, AssetName, UnitName, Total, Decimals, DefaultFrozen, URL, MetaDataHash) and mutable fields (such as Manager address, Reserve address, Freeze address and Clawback address). The information about the artistic right and the territory are stored in the parameters of the Algorand Standard Asset (ASA). The $\langle cc, ro, ri, terr \rangle$ values are stored in the ASA immutable fields, while the $valid\_from$ and $valid\_to$ values can be implicitly determined by the timestamp of the ASA asset transfer transactions. The percentage of the $share$ is determined thanks to total supply (equal to 100) and decimals (equal to 2) parameters. Total supply and decimals, used together, allow the rights holder to be free to assign any portion of the 100.00% of the $share$ to the cmo_1, another portion to another society cmo_2, avoiding, at the same time, the double-spending of the share. In this solution, the Metadata Hash field of every ASA stores the hash value of the IP private anagraphic data to verify such data's integrity later.

In this first approach, asset creation is (at least partially) centralized, while its management is decentralized. The asset is created by an issuer and a rights holder 2-of-2 multisignature account, which is the initial asset owner. We can have multiple issuers working independently (but consistently) from each other. These issuers are responsible for creating well-formed ASAs.

In an ASA, a clawback address specifies an account able to transfer a token it does not possess. In the reference solution, the clawback address corresponds to the rights holder blockchain address (i.e., the hash of its public key). In this way, at whatever moment, the rights holder can "take the control back" of the ASA and assign it to any other account. The structure of a right represented as an ASA is depicted in Figure 1.

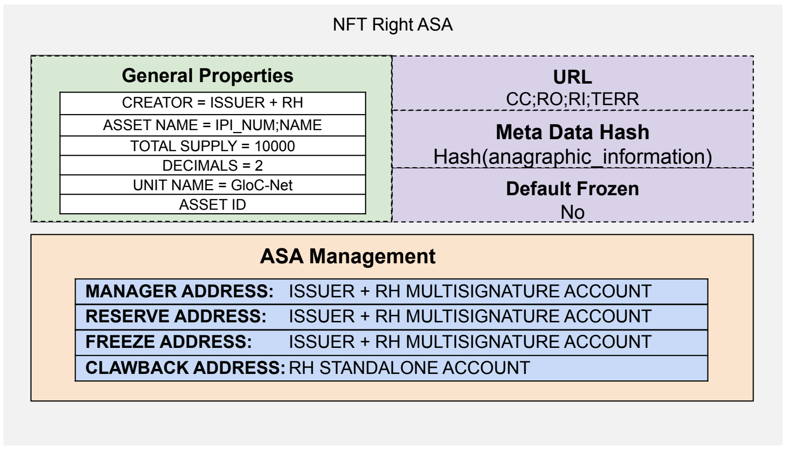
NFT Right ASA
General Properties
CREATOR = ISSUER + RH
ASSET NAME = IPI_NUM;NAME
TOTAL SUPPLY = 10000
DECIMALS = 2
UNIT NAME = GloC-Net
ASSET ID
URL
CC;RO;RI;TERR
Meta Data Hash
Hash(anagraphic_information)
Default Frozen
No
ASA Management
MANAGER ADDRESS: ISSUER + RH MULTISIGNATURE ACCOUNT
RESERVE ADDRESS: ISSUER + RH MULTISIGNATURE ACCOUNT
FREEZE ADDRESS: ISSUER + RH MULTISIGNATURE ACCOUNT
CLAWBACK ADDRESS: RH STANDALONE ACCOUNT

Figure 1: Structure of non-fungible ASA used in the reference solution.

Rights creation and management lifecycle. The reference solution involves three distinct categories of users: issuer(s), rights holders, and CMOs.

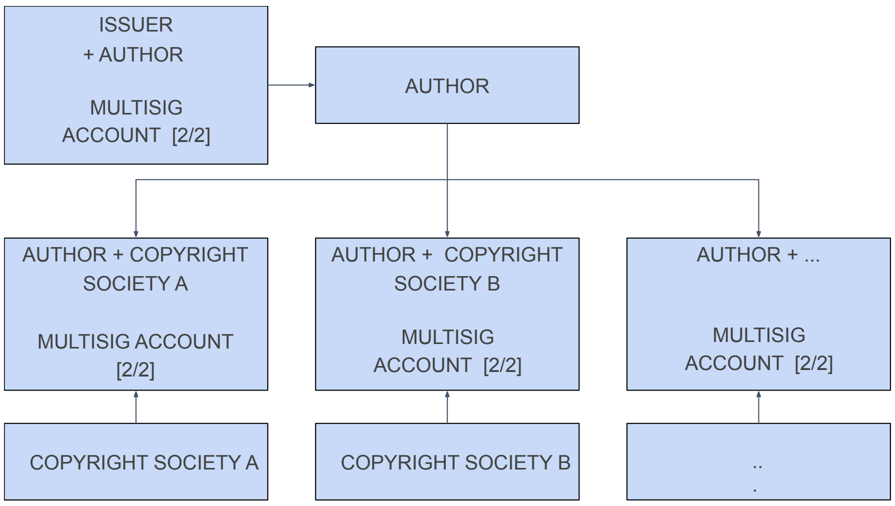
ISSUER + AUTHOR MULTISIG ACCOUNT [2/2]
AUTHOR
AUTHOR + COPYRIGHT SOCIETY A MULTISIG ACCOUNT [2/2]
AUTHOR + COPYRIGHT SOCIETY B MULTISIG ACCOUNT [2/2]
AUTHOR + ... MULTISIG ACCOUNT [2/2]
COPYRIGHT SOCIETY A
COPYRIGHT SOCIETY B
...

Figure 2: Roles and Architecture of the reference solution

The proposed approach works as follows, illustrated in Figure 2:

1. When the rights holder rh needs to assign the management of one of its rights, it first requests the issuer i to generate the corresponding *new* ASA with given $\langle cc, ro, ri, terr\rangle$
2. The multisignature associated with the i and rh creates the ASA.
3. The rights holder claws back the ownership of the ASA and transfer it to another multisignature account associated with collecting society cmo_1 and rh. This corresponds to assigning the management of the artistic rights associated with the ASA to the cmo_1, stipulating an agreement valid from the clawback transaction timestamp. Before receiving the ASA, the multisignature account, associated with cmo_1 and rh, has to opt-in to the asset, enabling its account to receive that token.
4. When the rights holder decides to re-assign the management of the artistic right to a different society (cmo_2), it claws back again the ownership of the ASA and transfers it to a cmo_2 and rh multisignature account. The values of the $valid\_to$ field of the previous agreement and the $valid\_from$ field of the new agreement are equal to the clawback transaction timestamp.

The multisignature accounts require 2 out of 2 signatures by each party composing it. This means that both parties must agree upon every transaction issued by the account (and, therefore, every on-chain activity). This prevents a single malicious party poisons the application with fake or duplicated data in new ASAs and prevents CMOs from opt-in ASAs they don't want to receive. Indeed, any attack requires both parties to collude.

The role of the issuer. The issuers are one or more entities in charge of maintaining the system's state valid. Their main goal is to issue new well-formed ASAs representing new rights. Moreover, an issuer can handle additional tasks. For instance, it can avoid double-spending on rights, checking, before issuing a new ASA, whether the ASA relative to an artistic right for that rights holder already exists in the system.

Depending on the governance requirements, an issuer can be managed:

1. by a single account;
2. by a multisignature account jointly managed by multiple accounts;
3. by a smart contract.

Limitations of the reference solution. In the reference solution, CMOs act as trusted issuers since they are already trusted in the centralized solution. A trusted issuer verifies off-chain that ASAs are well-formed and, ultimately, issues the new ASA representing the artistic right.

This solution does not prevent by-design the creation of multiple ASAs on-chain. CMOs are also trusted to issue new ASA if another representation of the same right does not already exist on-chain. If, by mistake, multiple ASAs appear on the blockchain representing the same right, the reference solution proposes considering the last ASA inserted on-chain as valid. All the previous ASAs are filtered out off-chain when reading from the blockchain as well as all the operations related to them.

3. Decentralize Solutions for Consistency Check

In the reference solution, we accept in the blockchain all the transactions. Through an off-chain decoding technique shared between honest users (i.e., only the last record on-chain is valid), we discard those transactions that do not respect the common rules. This requires a continuous audit of the blockchain or, if a peer disconnects for some time, scanning all the blocks of the ledger and executing consistency checks to discard invalid transactions. Executing these checks requires maintaining off-chain the entire state of the system and looking it up might represent a significant computational overhead if many users adopt it. Using a smart contract, encoding on-chain which assets are consistent with previous blocks of the ledger and filtering invalid transactions by design, reduces the computational and memory overhead for users.

With this motivation, this section describes some techniques to perform on a smart contract this type of verification. Note that since the rights were deployed on the Algorand mainnet, we will consider smart contracts with constrained resources in terms of computational power and input transaction parameters. Other platforms enabling smart contracts development (e.g., Ethereum [5]) would, however, benefit from such an approach because this decreases the amount of on-chain computation, reducing transaction fees.

We can distinguish between two types of techniques:

- Optimistic approach, where we do not prevent illicit asset creations but discourage them, thus, reducing them.
- Preventive approach, where we deny by design the creation of illicit assets.

3.1. Solutions Requirements

Before describing decentralized solutions to realize consistency checks on smart contracts, we need to analyze how to identify rights holders in the system and which are the bounds of Algorand smart contracts.

Identifying rights holder. In the IPI system, users are unequivocally identified with the primary key $base\_no$. The system centrally assigns this identifier. Let us suppose that a user on the blockchain has multiple identifiers (i.e., cryptographic key pairs) that uses to issue his/her rights. If there is no information on-chain to verify that these identities are linked, all consistency checks on the past blocks of the ledger will become ineffective.

Therefore, our analysis assumes that every user has only one identity on-chain and, eventually, linked identities declared on-chain. This requires the involvement of a trusted third party to associate the public keys to rights holders' identities in the real world. In this way, we have a centralized identity creation (i.e., the association between the public key and the rights holder's identity), but its management is decentralized (i.e., the user autonomously creates consistent assets with the past blocks of the ledger representing rights).

Algorand Smart Contract constraints. There are two types of Algorand Smart Contracts [6]:

- Stateful smart contracts which are designed to memorize arbitrary state variables according to the logic defined in the contract,
- Stateless smart contracts which consent to add to the blockchain only those transactions that respect the logic defined in the contract. In other words, they act as "filters" to the transactions that can be added on-chain.

In Algorand, each computation performed on a smart contract has a cost. The simplest operations, such as the arithmetic and the logical ones, have unitary costs. However, there are other computations, such as cryptographic ones, that have higher costs varying from some decades (e.g., sha256 costs 35) to thousands (e.g., ED25519 digital signature verification) of computation units [7]. On a Stateful contract, we can execute at most 700 computations, while on a Stateless contract, we can execute at most 20000 computations. Transactions invoking the contract can have a maximum size of 2KB in the case of a stateful contract and of 1000 bytes in the case of a stateless contract [6].

To exploit these instruments at the best, it is possible to invoke these two types of smart contracts simultaneously with a group of atomic transactions. Indeed, in Algorand, it is possible to create groups of atomic transactions that either commit all together or all abort if only one of those is invalid [6]. With this technique, we can combine these two types of smart contracts.

Stateful smart contracts memorize arbitrary state variables. The storage space is organized as a key-value map. However, their storage space is not unbounded. Indeed, a smart contract can store at most 64 key-value pairs if this represents a global state of the smart contract (i.e., global variables) and 16 key-value pairs per each user "subscribing" the smart contract (i.e., local variables). Each key-value pair is bounded to 128 bytes (see more details here [6]).

3.2. Optimistic approach

In this approach, any inserted information is assumed to be valid, but all the users can independently check the consistency of such information with the previous history of the ledger. If an inconsistency is detected, users can provide the pointer to the two pieces of inconsistent information to a smart contract that can invalidate one of the two according to a specific policy (e.g., delete the newest one). Moreover, we can penalize those issuers who have created inconsistent information through economic fees. Also, in this case, we do not prevent invalid transactions, but these are discouraged due to the ability of any user to invalidate them. We define this approach as *optimistic* because the system will eventually tend to a valid state.

For the sake of concreteness, let us analyze a concrete example with the support of Figure 3. A user u_1 wants to act as an issuer in the system, either on behalf of itself or for some other user. Before issuing assets, it has to (step 1) transfer an economic deposit to a smart contract that can be used to penalize it for an eventual malicious behavior. The smart contract (step 2) answers by issuing to u_1 a token, enabling it to create assets. Then, u_1 creates two assets a_1 and a_2 (steps 3 and 4), representing the same right for the same set of territories $\langle cc, ro, ri, terr\rangle$ for the same user. Referring to Figure 1, this means that a_1 and a_2 corresponding URL and clawback address fields of the token. Since a_2 has been created later, double spends the right and the order among the two assets is given by the asset id field, which is a unique counter in the blockchain.

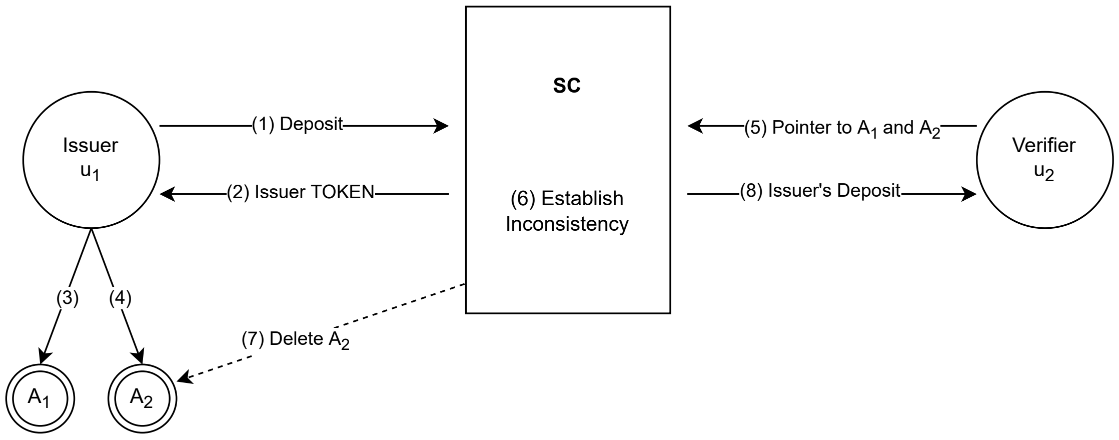

Figure 3: Steps of the optimistic approach

Suppose another user u_2 is auditing the blockchain to eventually detect inconsistencies. It notices a_1 and a_2 and that a_2 is an invalid asset. Therefore, (step 5) u_2 sends a transaction to the smart contract with two pointers to a_1 and a_2. The smart contract establishes (step 6) that those are inconsistent by 1) verifying that URL and clawback fields correspond and by 2) verifying that a_1 asset id is lower than a_2 one. If the verification succeeds, it will (step 7) burn a_2 and (step 8) transfer u_1's deposit to u_2.

3.3. Preventive approach

In this approach, only a smart contract is enabled to add new information to the blockchain. The users send transactions to the smart contract, "asking" it to add this new information. If the requested information is consistent with the past blocks of the ledger, the smart contract adds it. Otherwise, it denies its addition. In other words, a smart contract will add new assets representing user rights if and only if these are not already present in the blockchain.

The smart contract stores a succinct representation of the state of the ledger. The state represents the issued ASAs of the rights on-chain. If a user wants to create a new rights representation on-chain, he/she provides to the smart contract a proof, computed off-chain, proving that that right is not comprised in the state.

The smart contract can index storage space of different dimensions depending on the technique adopted to represent the state succinctly. In this section, we evaluate several of these techniques. In other words, this means that, with some solutions, we can check the consistency of a higher number of combinations of rights with respect to others. However, the higher the indexed space by a smart contract, the higher the number of computations might be, making some less powerful solutions more convenient in some cases.

The same flow of actions is performed for all these techniques. Indeed, user u_1, willing to act as an issuer, sends proof that the asset a_1 it wants to create is not registered in the contract state. If the proof is valid, the smart contract updates its state, and, finally, it issues a_1. If u_1 tries to create a_1 again, it will not be able to provide valid proof of the contract.

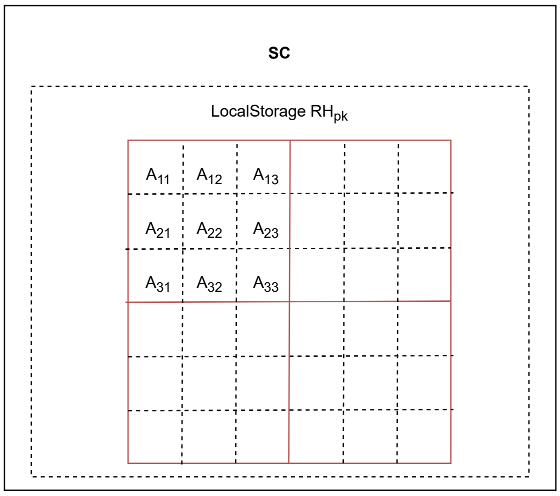

Figure 4: BitMatrix representation of personal assets. SC indicates the stateful smart contract.

3.3.1. BitMatrix solution.

The simplest solution represents the application state registered on the ledger as a matrix and stores it in the smart contract. Every element of the matrix A_{ij} represents a possible combination of cc, ro, ri and $terr$ and it has a boolean value. If true, a right has already been created; otherwise, its value is false. When a user wants to create a new asset, the smart contract verifies in the matrix whether the cell A_{ij} corresponding to the combination of cc, ro, ri and $terr$ has a value of 0. If so, the smart contract toggles A_{ij} value to 1 and creates an asset equal to the one of Figure 1. Note that, in this approach, the smart contract verifies by itself that the rights have not already been created without any proof computed off-chain by the user.

To enhance the storage space available in Algorand smart contract, we use a type of memory defined as "local". Every user has its local storage allowing the smart contract to scale with an increasing number of subscribing users. We recall that the local storage can host 16 local variables, each 128 bytes long and structured as key/value [6]. At least 1 byte must belong to the key and the value. Hence, if we use 1 bit for each cell of the matrix, the maximum number of addressable computations is $16 \times 127 \times 8 = 16256$. The number of computations to update or verify the matrix is constant.

3.3.2. Merkle Tree solution.

The BitMatrix solution needs a few on-chain computations but does not build or verify an elaborate succinct representation of the state. Indeed, the state is represented in its totality, directly saving a matrix and keeping count of the created rights in the storage space. However, storage space on-chain is limited. In this solution, we aim to represent a bigger matrix of created rights with a succinct representation of it in the smart contract. To achieve this goal, we perform more computation on-chain to verify a proof computed off-chain by users.

In particular, we memorize in the local variables of a user more root hashes of more Merkle

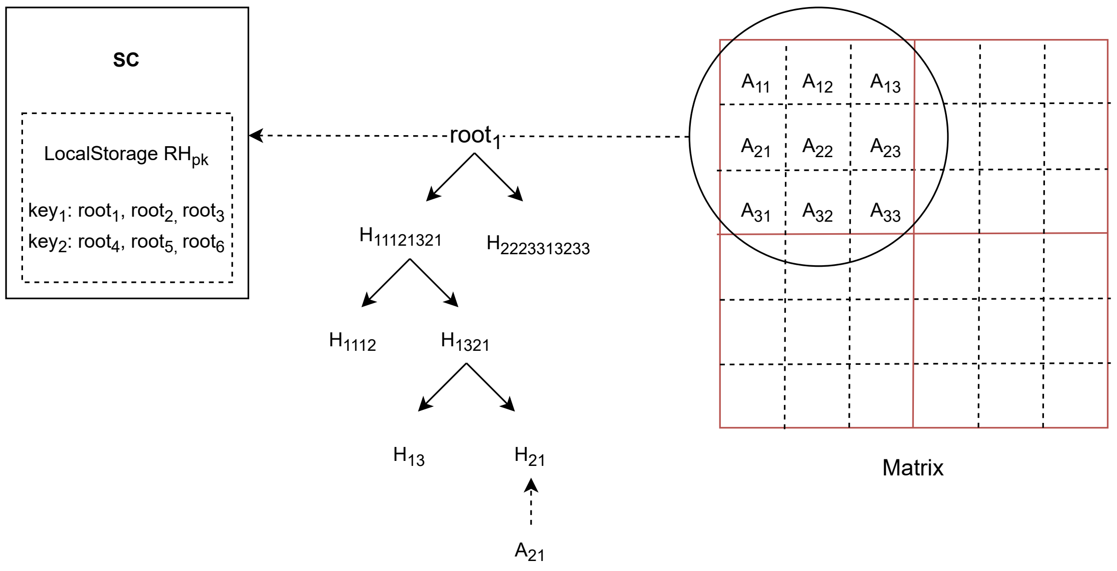

Figure 5: Merkle Tree representation of personal assets. SC indicates both the stateful and the stateless contracts combined.

trees. The leaves of these trees represent a combination of cc, ro, ri and $terr$. To update its state, the rights holder must provide the smart contract a proof that the leaf corresponding to the requested asset is set to a 0. If the proof is correct, the smart contract toggles the leaf to 1, computes the new root hash, stores it in a local variable and issues the new ASA.

The depth of the tree is bound by the smart contract constraints discussed in Subsection 3.1. In particular, this solution combines a stateful and a stateless smart contract to cooperate in the verification. The stateless smart contract verifies the proof's correctness and the stateful smart contract verifies that the root hash passed in the proof is stored in its storage space (in particular, in the local storage of the user).

One constraint of this approach is the transaction size invoking the stateless smart contract. It corresponds to 1000 bytes and it carries the Merkle proof. The maximum number of verifiable nodes is $1000 \times 8/256 \simeq 30$. Since for each level of the proof, we need 2 nodes, the maximum number of addressable nodes is 2^{15}. Since each root hash is 32 bytes long, we can store at most 3 root hashes in each local variable and verify $3 \times 16 \times 2^{15} \simeq 1.57$ million assets.

3.4. Discussion

Solutions comparison. Table 2 summarizes the pros and cons of every solution to draw the final results of this paper. In general, all the solutions do not appear difficult to implement in practice. The Merkle Tree solution requires a moderate effort to be implemented. The reference solution has no mechanism to look for inconsistencies in the ledger but it relies on off-chain encoding techniques. If discouraging double spending is secure enough to represent tokens that must be consistent with each other, we suggest an optimistic solution. If we do not need to represent a large set of tokens but we want to deny double spending, we can use the BitMatrix

Reference Solution	**Optimistic Solution**	**BitMatrix Solution**	**MerkleTree Solution**
+ easy to implement	+ easy to implement	+ easy to implement	- moderate to implement
- no on-chain lookup to the state of the system	+ efficient on-chain lookup to the state of the system (single transaction)	+ efficient on-chain lookup to the state of the system (single transaction)	+ efficient on-chain lookup to the state of the system (constant number of transactions)
	- no double-spending prevention by design	+ double-spending prevention by design	+ double-spending prevention by design
		- limited storage space (∼16k elements)	+ larger storage space (∼1.5M elements)
			- Not unbounded storage space

Table 2
Comparison among different solutions explored in the paper. "+" denotes pros of the solutions and "-" denotes cons.

solution because it is simpler to implement than the Merkle Tree one. Finally, the Merkle Tree solution allows the management of a more extensive set of tokens.

In our use case scenario, we handle a large set of right on-chain representations and, since, in the future, a large number of economic transactions might rely on this information, we aim to design a system that ensures the highest degree of security. For this reason, Merkle Tree is the solution that fits better for our scope.

Representation of the entire set of rights. In our use case scenario, the possible combinations of cc, ro, ri and $terr$ is $16 \times 34 \times 26 \times 220 = 3111680$. The solutions proposed in this section do not verify all the combinations required in our use case.

We can adopt a simple workaround to increase the addressable space for rights holders. In particular, every rights holder starts managing a primary identity Pk_{pr}, associated with its real-world identity, controlling more derived identities. Each derived identity refers to a creation class. Thus, we indicate the derived identities with $Pk_{cc_1}, Pk_{cc_2}, ..., Pk_{cc_{16}}$. Each derived identity must be uniquely associated with only one primary identity. In this way, for every cc the possible combinations of ro, ri and $terr$ are $34 \times 26 \times 220 = 194480$. Hence, the Merkle Tree solution is sufficient to verify all the combinations.

Every time Pk_{pr} invokes the smart contract SC to create a new ASA, the contract verifies that:

1. Pk_{cc_i} is associated *only* with the tuple (Pk_{pr}, cc_i),
2. The element in matrix of Pk_{cc_i} corresponding to the specified $\langle ro, ri, terr\rangle$ tuple is set to 0.

Property (2) is guaranteed as described in the previous subsections. To enforce Pk_{cc_i} to be associated only with one tuple (Pk_{pr}, cc_i) (property (1)), SC verifies that Pk_{cc_i}'s local variables in another smart contract SC_{id} are exactly (Pk_{pr}, cc_i). SC_{id}'s main goal is to maintain unique associations between every Pk_{pr} and all its derived identities. A derived identity can "subscribe" (i.e., opt-in [6]) SC_{id} only once, it cannot edit its state and it cannot "unsubscribe" (i.e., close-out [6]) SC_{id}. To handle fewer secret keys, which can represent a security issue if one of those gets lost or compromised, Pk_{cc_i}, can authorize Pk_{pr} the send transactions on its behalf. This mechanism on Algorand is called Rekeying [6].

4. Related Work

Copyright management on the blockchain is studied in the literature from various perspectives.

Audius [8] is a decentralized music streaming platform that allows anyone to publish and listen to music without paying subscription fees. Each creator can manage its revenues independently, unlike centralized systems. AUDIO, an ERC-20 token [9] that is also exchangeable on other blockchains (e.g., Solana [10]), powers the token economy, the transactions are held on Solana [10] and files corresponding to operas are hosted on IPFS [11]. Also, several other studies [12][13] describe how to design distributed streaming platforms based on blockchain. Fighting piracy using watermarks and verifying them with the information in the blockchain is the purpose of other approaches, such as [14][15]. Our study is based on the approach devised by SIAE, a relevant actor in the centralized solution, that wants to provide all the ingredients to support the transition from the centralized to a decentralized architecture without overturning the current system. This requires storing on-chain the information about artistic rights and agreements of rights holders with CMOs.

In general, smart contracts filter out the transactions that lead an application to an invalid state. In our case study, we filter out those transactions that double spend rights or create not well-formed tokens. Therefore, we implement consistency checks on-chain for our application domain. This type of inspection is present in the literature for different use cases. [16] describes a blockchain-based system for managing the land registry in India. Real estates are stored as assets within the blockchain; however, consistency verification with the rest of the system is provided by a centralized entity. Zakhary et al. [17] propose a global asset management system that unifies permissionless and permissioned blockchains. In the proposed system, a governmental permissioned blockchain authenticates the registration of end-user assets through smart contract deployments on a permissionless blockchain. Our approach removes any centralization point when we want to manage a finite set of digital assets not linked to assets in the physical world.

5. Conclusion and future works

In many application contexts, we need to guarantee that information inserted on-chain is consistent with the previous history of the ledger. The usual approach to handle this problem is to consider valid only the last record inserted or to trust a third party to perform the validation. In this paper, we have investigated how to verify consistency in a decentralized way, through on-chain smart contract computation.

We have shown how this problem affects real-world use cases, such as copyright management, particularly focusing on a first implementation sponsored by SIAE on Algorand mainnet. However, the described decentralized solutions for addressing this problem can check the existence of a limited number of identity attributes. Not controlling a very large set of assets may represent a limit in the adoption of such techniques. Indeed, it may not always be possible to find a workaround as we did in this paper. As a future work, we want to investigate further techniques, suitable for constraint contracts such as Algorand ones, to index a larger addressable space (around 2^{32} or 2^{64}) of elements. Furthermore, we plan to implement the analyzed solutions to verify consistency on-chain to evaluate their performances in terms of transaction fees and network latency.

References

[1] K. Ito, M. O'Dair, A Critical Examination of the Application of Blockchain Technology to Intellectual Property Management, in: Business Transformation through Blockchain: Volume II, Palgrave Macmillan, Cham, Switzerland, 2018, pp. 317–335. doi:10.1007/978-3-319-99058-3_12.

[2] SIAE, Italy's largest collective management organisation, represents authors' rights as digital assets managed on the Algorand blockchain, 2023. https://www.algorand.com/resources/ecosystem-announcements/siae-launches-4-million-nfts-on-algorand-for-creators [Online; accessed 26. Jan. 2023].

[3] F. Mogavero, I. Visconti, A. Vitaletti, M. Zecchini, The blockchain quadrilemma: When also computational effectiveness matters, in: 2021 IEEE Symposium on Computers and Communications (ISCC), 2021, pp. 1–6. doi:10.1109/ISCC53001.2021.9631511.

[4] Y. Gilad, et al., Algorand: Scaling Byzantine Agreements for Cryptocurrencies, in: Proceedings of SOSP '17, 2017.

[5] G. Wood, et al., Ethereum: A secure decentralised generalised transaction ledger, Ethereum project yellow paper 151 (2014) 1–32.

[6] Algorand Developer Docs - Algorand Developer Portal, 2023. https://developer.algorand.org/docs [Online; accessed 13. Feb. 2023].

[7] Algorand, Operational Cost of TEAL Opcodes, 2021. URL: https://developer.algorand.org/docs/features/asc1/teal/#dynamic-operational-cost-of-teal-opcodes.

[8] Audius - Empowering Creators, 2023. https://audius.co [Online; accessed 28. Feb. 2023].

[9] M. Shirole, M. Darisi, S. Bhirud, Cryptocurrency Token: An Overview, in: IC-BCT 2019, Springer, Singapore, 2020, pp. 133–140. doi:10.1007/978-981-15-4542-9_12.

[10] A. Yakovenko, Solana: A new architecture for a high performance blockchain v0. 8.13, Whitepaper (2018).

[11] J. Benet, IPFS - Content Addressed, Versioned, P2P File System, arXiv (2014). doi:10.48550/arXiv.1407.3561. arXiv:1407.3561.

[12] Y. Zeng, Digital music resource copyright management mechanism based on blockchain, in: 2020 3rd International Conference on Smart BlockChain (SmartBlock), 2020, pp. 158–162. doi:10.1109/SmartBlock52591.2020.00036.

[13] A. Kim, M. Kim, A study on blockchain-based music distribution framework: Focusing on copyright protection, in: 2020 International Conference on Information and Communication Technology Convergence (ICTC), 2020, pp. 1921–1925. doi:10.1109/ICTC49870.2020.9289184.

[14] W. Peng, L. Yi, L. Fang, D. XinHua, C. Ping, Secure and traceable copyright manage-

ment system based on blockchain, in: 2019 IEEE 5th International Conference on Computer and Communications (ICCC), 2019, pp. 1243–1247. doi:10.1109/ICCC47050.2019.9064101.

[15] Z. Meng, T. Morizumi, S. Miyata, H. Kinoshita, Design scheme of copyright management system based on digital watermarking and blockchain, in: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), volume 2, IEEE, 2018, pp. 359–364.

[16] N. Gupta, M. L. Das, S. Nandi, Landledger: Blockchain-powered land property administration system, in: 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2019, pp. 1–6. doi:10.1109/ANTS47819.2019.9118125.

[17] V. Zakhary, M. J. Amiri, S. Maiyya, D. Agrawal, A. E. Abbadi, Towards Global Asset Management in Blockchain Systems, arXiv (2019). doi:10.48550/arXiv.1905.09359. arXiv:1905.09359.