Securing the Art Market with Distributed Public Ledgers*

Marino Miculan and Daniel Tosone

MADS laboratory, Department of Mathematics, Computer Science and Physics, University of Udine
marino.miculan@uniud.it, tosone.daniel@spes.uniud.it

Abstract

The art market is an important scenario where many players (artists, buyers, sellers, brokers, etc.) act in a context with partial trust, and still it is crucial to ensure authentication and provenance of artworks. In this paper, we propose to store artworks information and ownership in *distributed public ledgers*. To this end, we present a distributed, scalable application which provides the main operations for operating on the art market. This application is composed by a *back-end* based on Ethereum ledger and the IPFS distributed file system, where the data is kept and manipulated by means of suitable Solidity *smart contracts*; and a *web-based front-end*. In this way, we obtain the robustness and scalability of Ethereum public ledger with the easiness of common web applications (or even mobile apps). The solution we propose could be applied to similar traceability contexts, where we have to deal with many partners with limited (or no) trust.

CEUR-WS.org/Vol-2580/DLT_2020_paper_4.pdf

1 Introduction

The art market, as we know it today, has just turned 140. In these years its value has constantly grown until it reached 67.4 billion dollars in 2018 [6]. Correspondingly, as the value increased, a series of problems have arisen that still remain without a solution. Asking to experts, there is a substantial consensus among the parties about the main threats to the reputation and functioning of the art market. Not all problems can be tackled through technological solutions (because they depend on social and economic aspects), but here we focus on two main issues.

The first one is *authentication and provenance*, by which we mean the assessment of the identity and the history of an artwork, based on the changes of its ownership. Provenance is crucial to assess the right of selling and the authenticity of an artwork. The second issue is *lack of transparency*. The art market is more opaque than others: many aspects, like the conditions of the selling, the artwork's value, its history, etc., are often unclear, and information are usually stored in private registers making difficult to check truthfulness of what is declared. In addition, the Italian legislation requires compliance with a series of measures concerning the circulation of artworks [2, art. 10], resale rights [1] and sales certificates ([2, art. 64], [3]).

Currently there is no valid solution to these problems. Improvements in provenance traceability have been made with the possibility of storing business records on computer supports [4]. However, the adoption of the necessary means is proceeding very slowly and, as a result, much documentation about artworks is still on paper. In any case, solutions supported by centralized databases do not meet the requirements of transparency we are looking for and, even in digital format, it is very difficult to build a consistent history of artworks for the need to check many private and disconnected sources of information. Moreover, these solutions are more vulnerable to attacks to availability and integrity due to the presence of a single point-of-failure (data not duplicated between databases) and that they rely of the underlying platform and the staff operating on it (e.g., an infidel database administrator can modify data for personal interest).

*Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0). Supported by the UniUD PRID *ENCASE* and Italian MIUR project PRIN 2017FTXR7S *IT MATTERS* (Methods and Tools for Trustworthy Smart Systems).

To address these problems, in this paper we propose to store artworks information in *public ledgers*, such as Bitcoin's *blockchain* and Ethereum's ledger. In particular, we design and develop an application which allows the principal actors of the art market to perform and verify the main operations concerning the commercial lifecycle of artworks. Differently from other proposals, all the process can be executed inside the public ledger, and hence fully open and verifiable. This application is composed by a *back-end* based on Ethereum ledger and the IPFS distributed file system, where the data is kept and manipulated by means of suitable Solidity *smart contracts*, and a *web-based front-end*, which allows also non-experts to access the system's functionalities and without the need of running a local Ethereum node. In this way, we retain the robustness and scalability of public ledgers with the easiness of web applications (or even mobile apps).

The rest of the paper is organized as follows. In Section 2 we detail the problem under consideration. The solution we propose is described in Section 3, and experimental results are reported in Section 4. Finally in Section 5 we draw some conclusions and outline future work.

2 Problem description and analysis

In this section we briefly describe the main actors of the system under development, their roles, and the main requirements that a solution has to satisfy.

The core object to manipulate is the **artwork**. Each artwork must be given a globally unique identifier. At every time, we must be able to know the current owner of the artwork, as well as the history of previous ones (i.e., the provenance). Artworks can come with metadata (e.g., date of creation, picture, size, technique, etc.), which must be kept in the system as well.

The principal actors, and the functionalities they must be provided with, are:

Artists, capable to: register their artworks (obtaining unique identifiers for them), enabling a certified tracking of provenance for collectors; sell their creations on the platform directly or through a broker receiving the remuneration automatically at the moment of selling; check the correctness of payments.

Collectors, capable to: buy and sell artworks, checking authenticity and provenance; ask for authentication of artworks they own.

Brokers, capable to: sell the artworks under their custody, with automatic and verifiable distribution of selling percentages.

Authenticators, capable to: see all requests on the platform and authenticate artworks. The list of works they authenticate must be open to consultation by all users.

A good solution should satisfy the following requirements:

Transparency: As much information as possible has to be accessible without filters, and it also ought be possible to check the correctness of processes. Therefore, not only information but also operations and procedures must be registered and publicly available.

Integrity: must be guaranteed in a "no trust" context in order to preserve the value of data, in particular that related to provenance and authentication of artworks.

Availability: Information should be available to consultation at any time. In particular we do not want to rely on any central authority for keeping data available. The system must also be resilient to failures and sufficiently scalable.

Usability: The functionalities of the system must be usable by users with expertise in the applicative domain (as listed above), but with little o no knowledge of the technological aspects behind the implementation.

3 Solution proposal

To meet requirements described in Section 2 we will design our system according to a strong decentralization principle, in two ways. First, we eliminate the need of central authorities by distributing the control of the platform among his members. This is fundamental to guarantee a true transparency: as long as a central entity controls everything, the transparency ultimately boils down to trust this entity. Secondly, we distribute information and processing among a peer-to-peer network. This choice is useful to increase security: decentralization allows to avoid single points-of-failure and is also helpful in increasing the system scalability and availability.

Decentralization is obtained by storing data and procedures in a *distributed public ledger*. Since the seminal paper [14], many implementations of public ledgers have been provided. For the problem of this paper, we have considered three main alternatives:

Bitcoin [14] is the most decentralized and immutable, due to its large number of nodes and high hashpower involved. However, its programming language has a limited expressive power, hindering the possibility of defining generic programs.

Ethereum [8] keeps a high degree of decentralization and immutability but also includes the support for the definition of *smart contracts*, programs written in a Turing-complete language executed on different nodes in order to guarantee a correct agreement. They consequently inherit blockchain properties.

Hyperledger Fabric [5] is a framework for building and running *permissioned* blockchains, which aim to improve privacy and efficiency. However, the price is paid in lower decentralization; hence fault tolerance is reduced and transparency is limited by the need to trust the restricted group of blockchain maintainers [15].

After analysis, Ethereum has been chosen as the most suited for the purpose and therefore it has been used as core of the project. Beside immutability, forgery resistance, decentralization, it is more flexible than Bitcoin's blockchain and more transparent than permissioned blockchains.

However, we must be careful in the adoption of a public ledger for storing artwork exchanges. Although most users of the art market would ask for as much transparency as possible, especially on the ownership of artworks, there is also the need to protect artworks from possible thefts. With this in mind, not everyone wants to make public the possession of very expensive works or their purchase price. This is a problem because it makes very difficult to check provenance and the payments of resale rights and brokerage percentages.

With this in mind, we have opted for a trade-off. For each artwork, its owner's name will not be made public but only his address (a public key of 20 byte code with no personal data associated). The chain of the various addresses that have possessed a certain work is sufficient to keep track of the provenance and the right to sell; the very first owner is the artist, registered with his name. Regarding the purchase price, instead, the user is free to choose whether to make it public, helping to increase market transparency, or keep it private. The risk of discouraging investor participation is too high and it was preferred to accept this compromise in favor of participation in the process of enrichment of provenance data. We can however figure out that many investors are willing to share the purchase price when the amounts are not very high and that some consider the pseudonymity of the assigned address a sufficient guarantee to protect themselves from theft, thus making the purchase price public.

This solution leads us to introduce the role of the *administrator*. His task is to identify the actors and certify the correspondence with the Ethereum address they provide. In the case of artists, authenticators and brokers the association between address and identity will be known by the network. Instead, the identity of collectors is known only by administrator in order to

guarantee the fulfillment of legal obligations (the administrator always knows selling prices). His powers are limited to the minimum, as he does not have control on the exchange of artworks and cannot modify data on provenance and authentication.

3.1 General architecture of the system

The application is two-tier, structured in a front-end and a back-end, as shown in Figure 1.

The front-end consists of a set of web pages (one for each player plus an index page) through which it is possible to insert and display data on Ethereum. These pages integrate a part in JavaScript which is responsible for rendering, user interaction and interaction with back-end. The communication with the blockchain takes place by means of a JSON-RPC to a node on the Ethereum network that actually executes the transaction requested by the front-end.

The back-end is mainly constituted by the Ethereum blockchain, plus two auxiliary modules. The first is IPFS [7], a distributed file-system needed to store additional information about artworks, like their images (storing them directly on Ethereum would be too expensive). The second is a traditional database to store data that must be private. Ethereum is indeed a public blockchain and everything saved on it is publicly available to everyone. The Ethereum part is organized in modules based on the functionalities offered. For each module there is a smart contract defined in Solidity [9]. The modules and the respective offered functions are depicted in Figure 2.

The main contract in this scheme is represented by module *Token*. It contains the information and operations about registration, exchange and tracking of artworks. It is not called directly but needs to be called via the other modules that do specific checks and additional routines before using the central module to perform the requested action.

3.2 Back-end

The architecture of the back-end is in Figure 2. We will now present shortly the main modules.

Token: Artworks are associated with *tokens* with a unique ID following the ERC-721 standard [10], commonly accepted for non-fungible tokens on Ethereum. Each token will be associated with the artwork description: title, author, year, medium, dimensions, authentications and hash of the information needed to its identification. For convenience, this information is stored in a P2P file system (see "Storage of additional information" and Section 3.4 below). This module offers also a simplified interface on Etherscan chain explorer for provenance tracking.

ArtistMng: In addition to the functions displayed in Figure 2, through this module the administrator can indicate for every artist if he still holds the resale right and the artist can state the address authorized to withdraw the funds after his death.

Authentication: The system requires collectors to obtain two authentications in order to register an artwork. By requesting authentication they upload the image on IPFS and the request on Ethereum. Registered authenticators can then attach their authentication (possibly after having examined *de visu* the physical object).

BrokerMng: It enables to entrust artworks to a registered broker, also specifying the reserved commission. It performs checks about possession, before calling the function in *Token*.

Marketplace: It implements the sale function. Every player must interact with this contract to transfer property of an artwork. Both public and private sales are provided, but the two cases are quite different in terms of actions performed by the software. In the case of a public sale, the exchange of token's possession is done simultaneously with payment transfer.

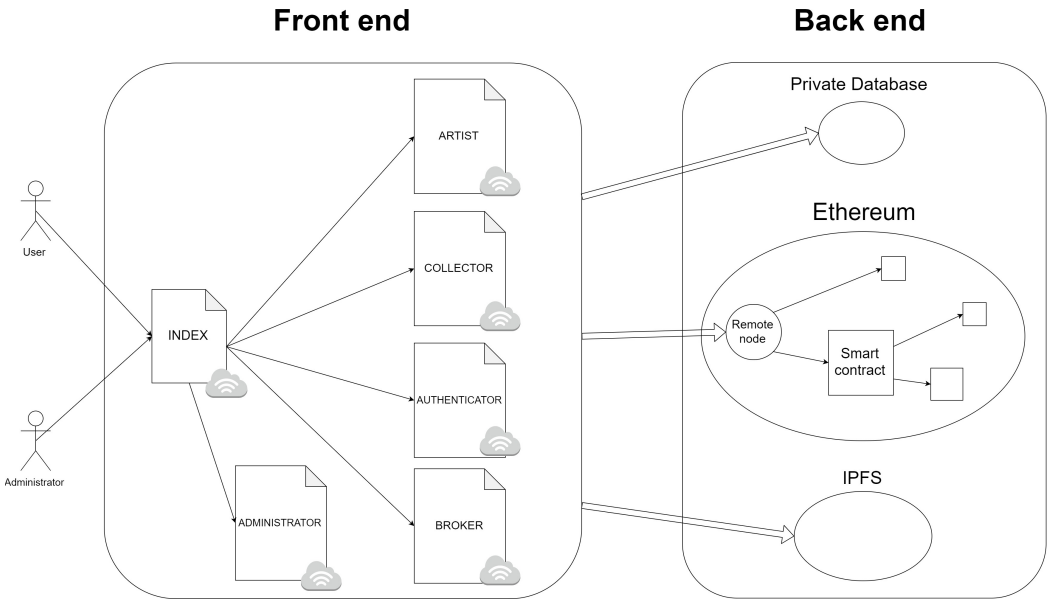


Figure 1: Architecture of the application.

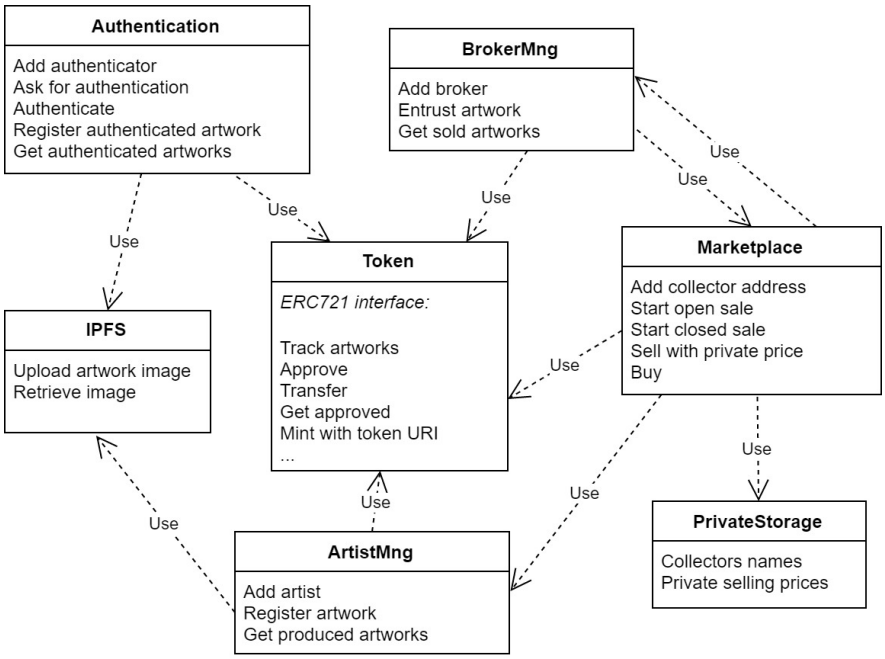


Figure 2: Back-end division in modules.

This is done by the contract code in a decentralized and atomic way. Doing so, if any error occurs during the transfer (e.g. for latency or prior sale) everything is undone and the previous state restored. The contract will always behave according to its programming, due to its execution on the blockchain. On the other hand, the private sale does not ensure the same security properties, requiring the seller to receive payment outside Ethereum. Sale prices and correspondence between addresses and collectors' identities are kept in a private storage which can be accessed by the Administrator to verify correctness of sales outside the public ledger.

Storage of additional information: The system must memorize information for physical identification of the artwork. In the current version this consists of a digital (high-resolution) picture of the artwork, but it can be extended with other physical-chemical data. To store this data we have three possible choices. The first option is to use a traditional database but, in that way, we would have drastically decreased transparency, decentralization and security of the system. The second option is to store artwork images directly in the Ethereum blockchain, but this is quite expensive, as it would force the user to pay about € 300 for every 300 kB[1]. Finally, we can adopt a distributed storage service, like *IPFS*, which is based on a peer-to-peer network. In IPFS, contents are referenced by their *CIDs*, which are mainly constituted by the content hashes. The service uses a DHT (*Distributed Hash Table*) in two occasions: firstly to find a peer that hosts the content requested, secondly to obtain its current location. IPFS allows to keep economic costs down while maintaining the same properties found in blockchain. It is transparent (everyone can view the file having its CID), decentralized (every peer can store a copy of the file without asking for authorization) and verifiable (hash will not match if the content is modified). The main difference with Ethereum is that here availability is not guaranteed: it is up to the nodes will to keep a copy of the file. However, in a truly large scale peer-to-peer setting it is reasonable to assume that there are always some active nodes hosting the files (e.g., the administrator's), and as long as there is at least one node hosting the file, anyone can be sure that the content has not been modified.

For these reasons, we have adopted IPFS to save the artworks' images. The hash, needed to refer the file, is saved in the artwork description on Ethereum so that it cannot be changed, thus guaranteeing integrity and uniqueness of the image associated with a token. On the other hand, the check that an image is referred to by only one token is implemented by the front-end and ultimately is upon artists and authenticators.

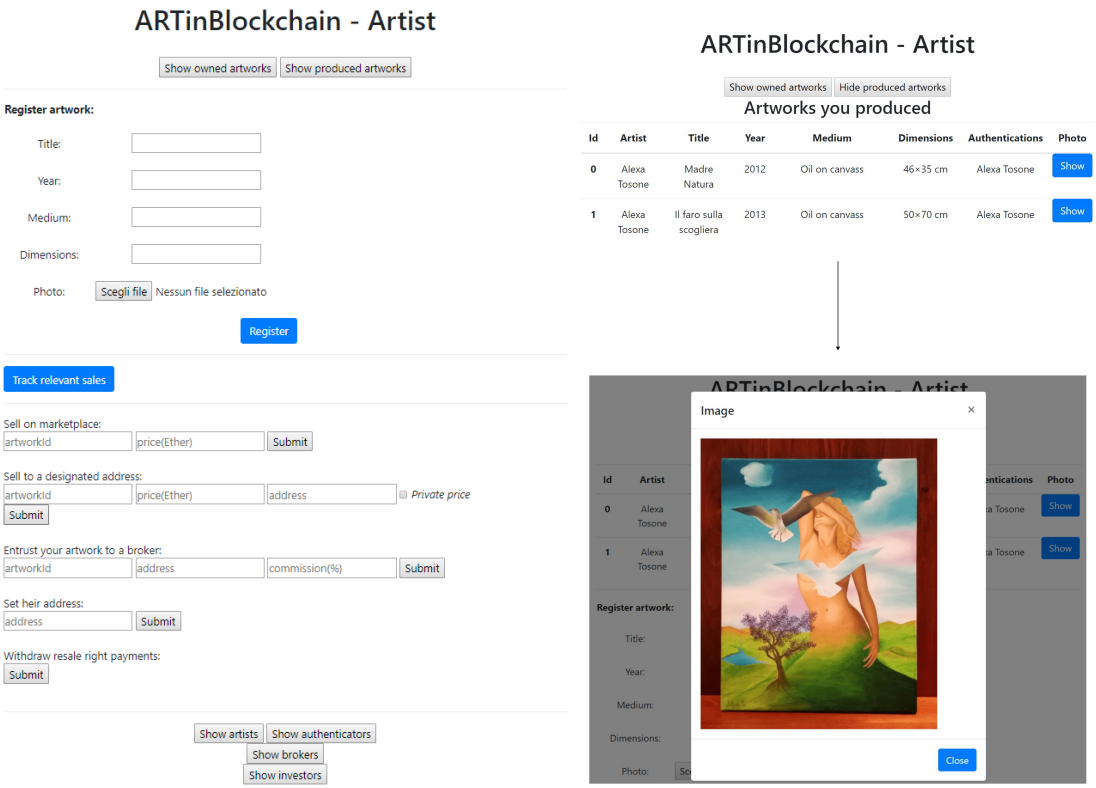
3.3 Front-end

Metamask is a browser extension that acts as a wallet where the private key associated with the public address is kept and used to sign transactions [12]. In order to check and execute smart contracts the user could run a local Ethereum node, but in general this could be too resource expensive for the generic user. Hence, to relieve the user of this burden, Metamask includes Infura[2], a provider of remote nodes. At the download, Metamask provides the public address that will be used to interact with the platform. It also provides a simple interface to see the transactions costs, the transaction history with details and estimated confirmation time. Metamask will always send a notification when a transaction is accepted. Every time a user wants to access the application he should firstly login to Metamask. At this point, the index web page automatically redirects the user to his page based on the address stored in Metamask.

Interface overview: Each web page contains buttons and forms to submit transactions to

[1] Every transaction on Ethereum costs a certain amount depending on complexity and network traffic to be processed [16]. The reported calculation is based on tests done on 14.09.2019 at 14:08 UTC.

[2] https://infura.io/docs



(a) Full visualization

(b) Produced artworks' details

Figure 3: Artist's page.

back-end. In addition to dealing with rendering and invoking back-end functions, the code is also responsible for uploading files to IPFS using the related APIs when an artwork is registered by an artist or when a collector asks for authentication. Pages are all constructed upon the same model, removing or placing the functions required by the specific role. An example is provided by Figure 3. To avoid interrogating the back-end when the negative response can be predicted beforehand, the front-end performs some legitimacy checks on operations (e.g. on the ownership of the artwork). It is important to notice that front-end is intended to help the user in interaction but is not mandatory: in fact, users can interact directly with Ethereum and IPFS. In this case, however, it is necessary to run a node and a peer (even remote) to bypass the interface provided. Compliance with the constraints is in any case guaranteed by the execution of smart contracts.

Figure 4 shows the simplified interface for provenance tracking. All transfers of tokens are mapped in *Token* contract and can be easily retrieved from this page. Selection can be done based on transaction hash, seller, buyer and tokenId. The address—identity association for artists, brokers and authenticators can be retrieved with the dedicated buttons at the bottom of each personal page. Even in this case, it is not mandatory to rely on external chain explorers, such as Etherscan. Many other chain explorer are available and the same information is retrievable also running a proprietary Ethereum node.

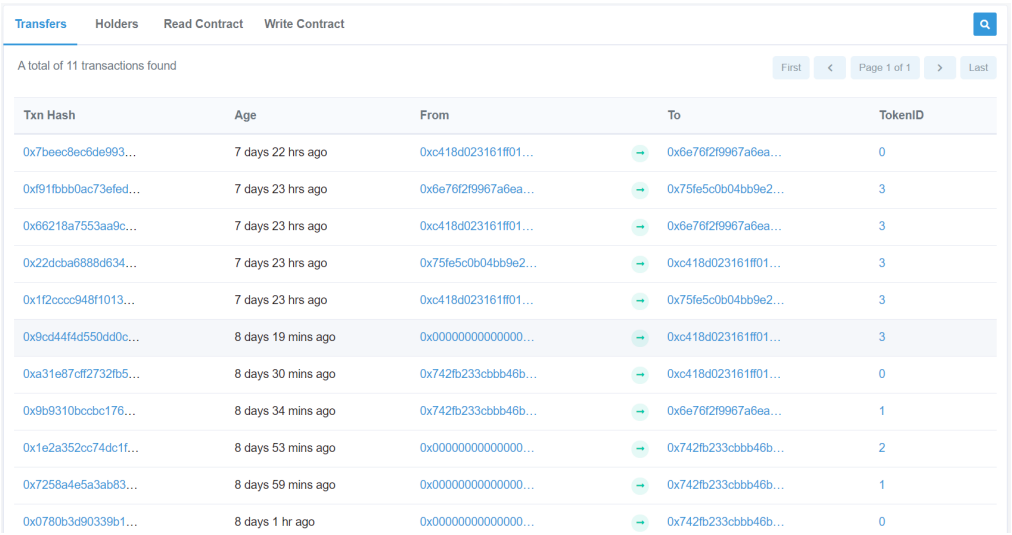


Figure 4: Provenance tracking.

3.4 Connection between real and virtual

Providing a robust link between the token on Ethereum and the physical artwork is crucial: without this step, the provenance data in the blockchain would not actually track anything. Moreover, we must ensure that a token will never be associated with more than one artwork.

The actual realization of this connection depends on the specific nature of the artwork, and can be implemented in several ways. A solution could be the application of a RFID tag on the canvas back, directly by the author or by an authenticator. New artworks can even use canvases with RFID tags directly woven into the fabric. Since a RFID tag does not need a battery to operate, it lasts basically forever and, if it is applied in such a way that its removal would indelibly ruin the artwork, we get a permanent reference to the corresponding Ethereum token. In this way, the buyer can easily check the provenance of the artwork, the associated information and its authenticity. He can also check that the seller has the right to sell it, comparing his signature with the public key of the owner registered on Ethereum.

It may be the case that authentications turn out to be wrong, after some time; this can be due to mistakes in the authentication process, or to malicious behaviours by the artist or the authenticator. In both cases, a new token is emitted and a new RFID applied next to the older one, marked as invalid. It is important to notice that the wrong authentication is recorded in public ledger forever, thus a dishonest behaviour would harm a user's reputation indefinitely.

3.5 Sequence diagrams

In this section we overview the workflow of the principal functions offered by the system.

Artwork registration by artist (Figure 5) is executed when an artist registers a new artwork in the system. In this diagram the user (artist) is already logged in the system, and interacts with the front-end, which is composed by a webpage and Metamask. The front-end interacts with the back-end, which is composed by IPFS, the Ethereum node which can be local or remote and the related smart contracts. The message Transaction registered is the event caused by the validation of the result from the execution of the called smart contract on Ethereum public ledger.

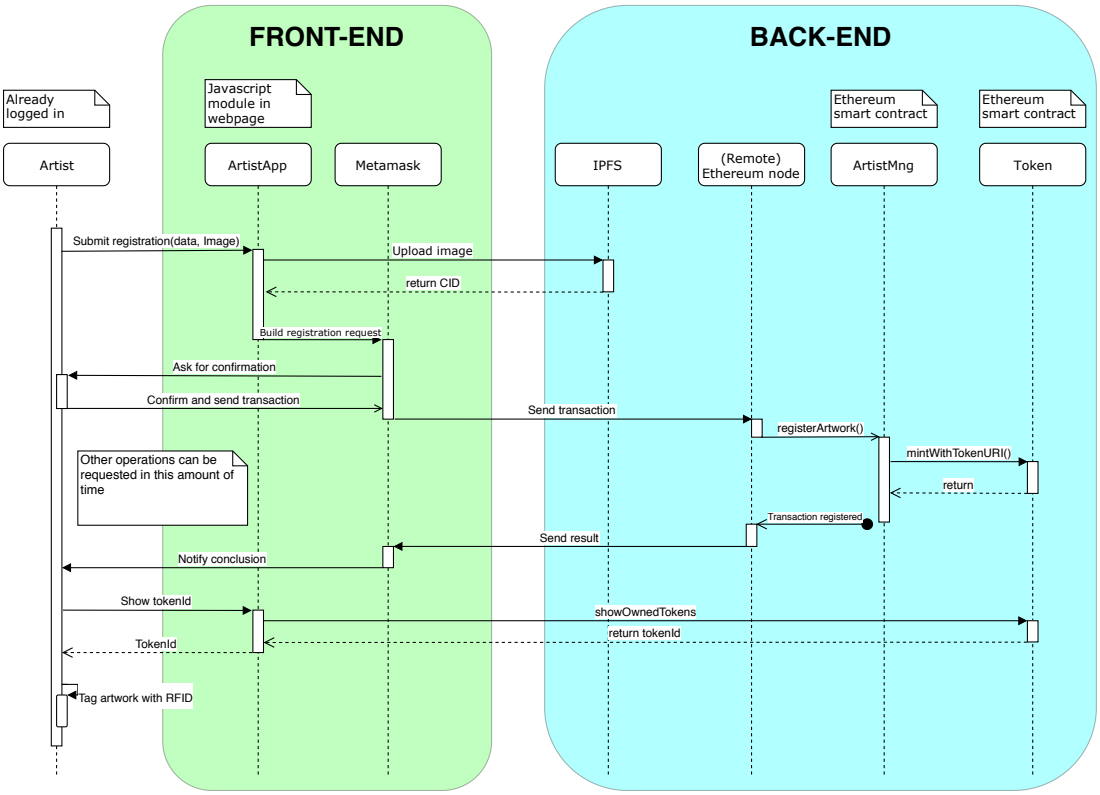


Figure 5: Sequence diagram for artwork registration by an artist. All requests yielding the execution of a contract follow the same pattern through Metamask and an Ethereum node; on the other hand, read requests are executed immediately. For sake of simplicity in the next diagrams we will omit the steps with Metamask and the Ethereum node, establishing a direct link between Javascript modules and Ethereum smart contracts.

Artwork registration by collector (Figure 6) is executed when a collector wants to register an artwork that he already owns. Before concluding the transaction, the collector has to wait for at least two authentications. Authenticators' webpage allows them to visualize authentication requests and related data (e.g., images), and to release a new authentication proposal or submit an existing one through Authentication contract.[3] The collector can check through his webpage when his request has enough authentications and then can call the registration method (which would fail if the request has not received enough authentications yet). Finally, the authenticator applies the RFID marked with the given token to the artwork.

Artwork buy and sell (Figures 7, 8) are the sequences executed when a buyer wants to acquire an artwork, and correspondingly a seller sells it. The diagram for the sell procedure covers both the public and private exchanges; clearly in the last case the value of transaction is not kept on the Ethereum ledger. This diagram shows the sequence from a collector's point of view but the same operation can be equivalently performed by an artist (through ArtistApp).

StartOpenSale works similarly to closed sales, but anyone can buy the artwork. Brokers can perform closedSale (for an entrusted artwork); these sales are saved by Marketplace contract.

[3]Authenticators are also expected to verify any pre-existing off-ledger authenticity and provenance documents, but this process is outside the scope of the present work.

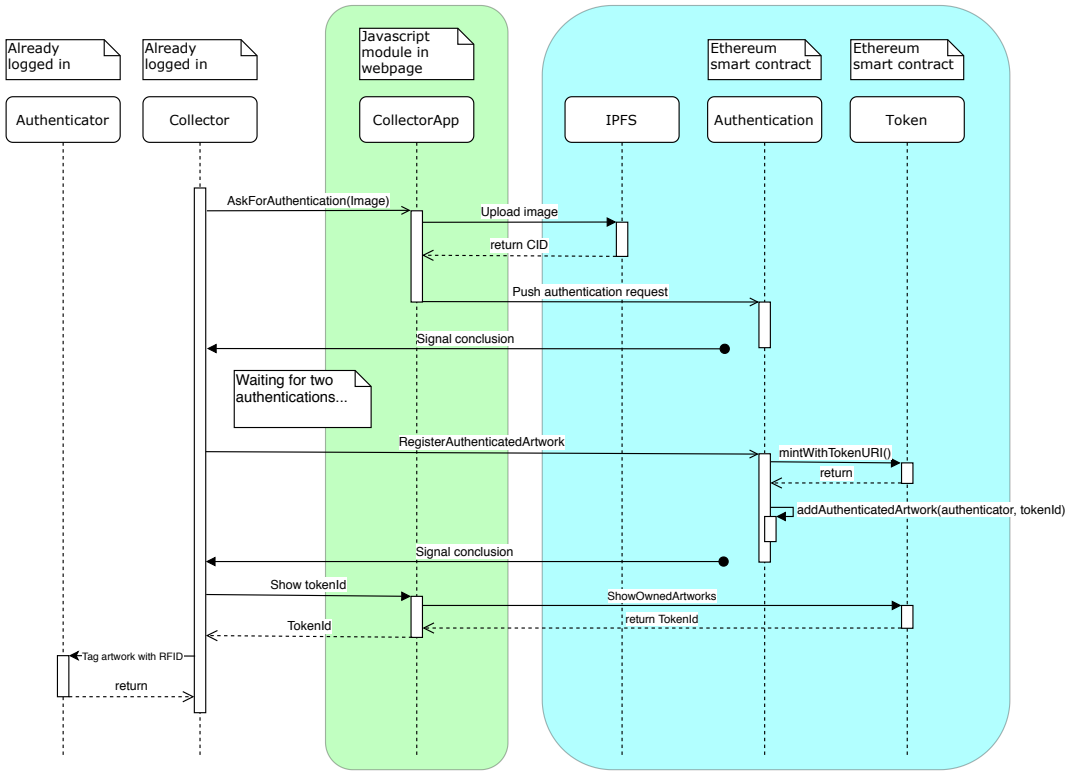


Figure 6: Sequence diagram for artwork registration by collector.

4 Evaluation and tests

All functional requirements from Section 2 have been satisfied (with the necessary feasibility conditions). Let us examine non-functional requirements.

Transparency: All data on Ethereum ledger is public. Information can be retrieved using the functions provided by the front-end[4], but a user can bypass the front-end and access the contracts history directly with Etherscan. Correctness of processes is verifiable by the distributed execution and the free consultation of source code on contract page (e.g. Marketplace). Moreover, both contract executions and member's operations are registered as transactions like the one in Figure 9. These transactions are forgery-resistant and non-repudiable, because signed with private keys. Not even the administrator can modify this data or avoid their distribution. It is thus possible for everyone to monitor actions on Ethereum with useful added security properties. The administrator will be consulted by artists and collectors only for checking payment correctness of sales with private price, thus reducing at the minimum the trustiness upon him.

Integrity is guaranteed by Ethereum itself, through the combined use of hashes, proof-of-work and distributed storage. As it is well-known, the asymptotic valid view of blockchain is the longest one. Therefore, in order to modify some data on the blockchain, an attacker should acquire and maintain for a sufficient time at least 51% of the total hash power of the network. An attack like this is proven to be economically and practically unfeasible.

[4]Source code is available at `https://gitlab.com/daniel.tosone/dlt-2020`. Links to Etherscan contracts pages are available in the repository description.

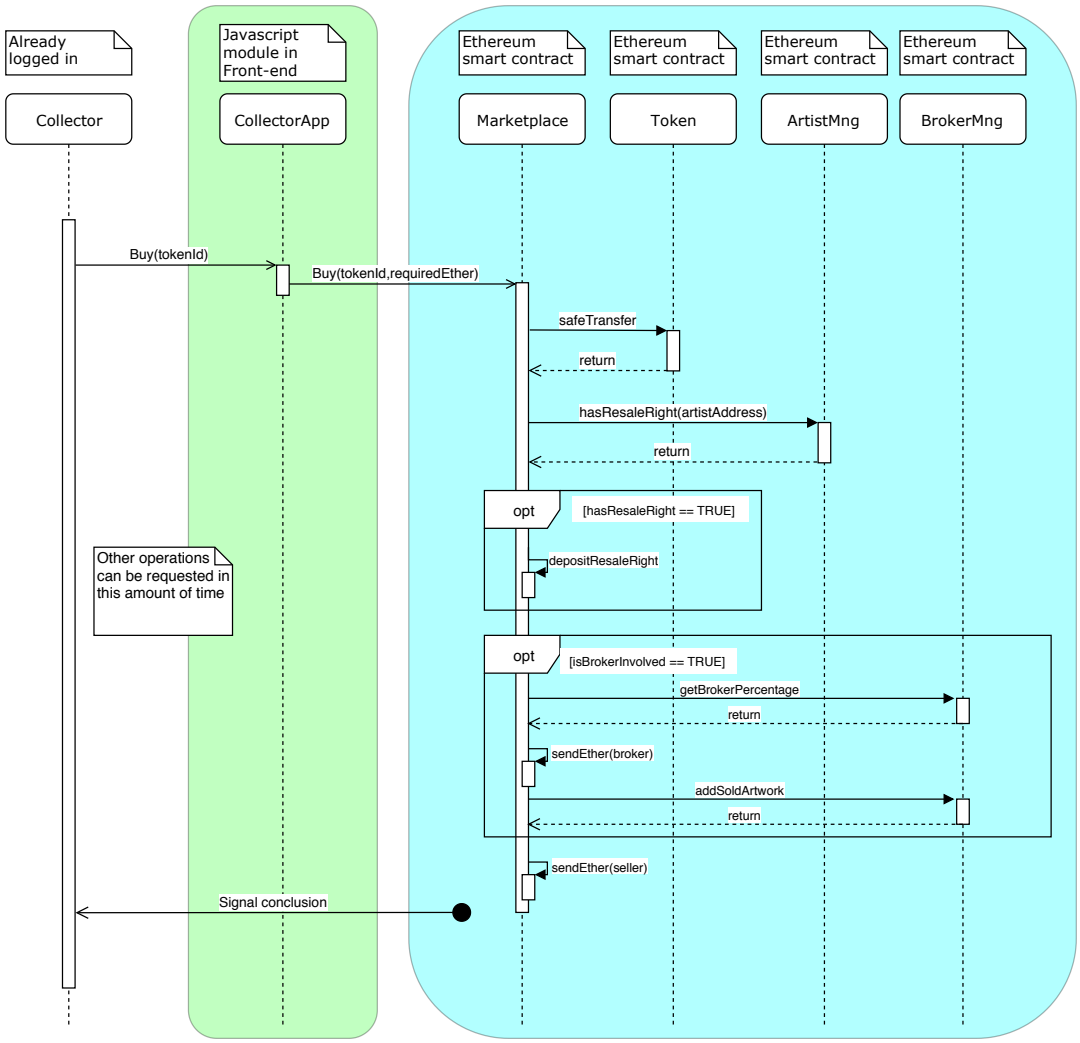


Figure 7: Sequence diagram for artwork buy.

Availability: Ethereum blockchain is replicated through thousands of peers (about 7000 nodes active, at the date). Failures of a portion of nodes is readily compensated by others. Protection against DOS attacks is guaranteed by the need to pay a sufficient fee in order to have transactions elaborated (growing as the complexity of operation requested increases). The attacker should then send enough transactions with the highest gas price to fill all nodes pools, and again the attack is economically unfeasible for widely used blockchains, like Ethereum.

A different argument goes for availability of images loaded on IPFS. To visualize an artwork image it is needed that at least one IPFS peer containing it is active. With a large IPFS network, this can be achieved with a good level of redundancy. Moreover, we can execute an IPFS peer on user's platform, as soon as he is interacting with the platform: when the user retrieves an image, the local IPFS peer automatically caches it in the browser, and as long as this peer is active other users can access this copy. Consequently, the degree of redundancy (and hence the availability) increases as the number of active users increases.

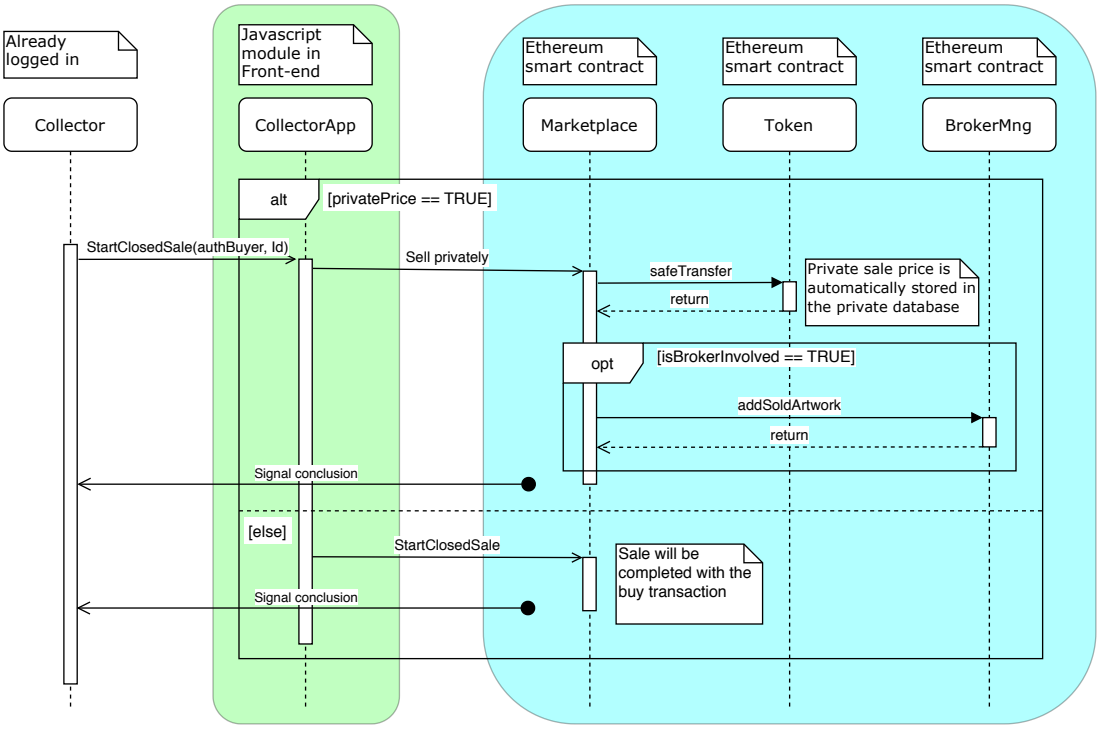


Figure 8: Sequence diagram for artwork sell.

[This is a Ropsten **Testnet** transaction only]

Transaction Hash:	**0x7beec8ec6de993de1a45e72e2ec9873fcd64d92324a057b408bf4ee5eeaffaa7**
Status:	Success
Block:	6415598 463055 Block Confirmations
Timestamp:	73 days 2 hrs ago (Sep-18-2019 03:54:43 PM +UTC)
From:	0x6e76f2f9967a6ea85f704a509d4f77ce5065c964
To:	Contract 0x4d89583d0eff1f5d424aab60d20c38e1f6719ab4 TRANSFER 1.8 Ether From 0x4d89583d0eff1f5d424a… To 0xc418d023161ff017ca66… TRANSFER 0.2 Ether From 0x4d89583d0eff1f5d424a… To 0xbd70d1adcd32261a8ad…
Tokens Transferred:	From 0xc418d023161ff01… To 0x6e76f2f9967a6ea… For ERC-721 TokenID [0] ArtToken (ART)
Value:	2 Ether ($0.00)
Transaction Fee:	0.000176391 Ether ($0.000000)

Figure 9: An example of transaction on Etherscan (a “buy” transaction through a broker).

Concerning scalability, we can say that the system is scalable (although no practical tests have been done yet), since each of the components of the system is scalable, or can be made as such. Ethereum has many entry points for transactions, Infura has a fleet of nodes managed through a load balancer, and the front-end and the private database can be replicated as needed, with or without a load balancer. In any case, the front-end can be bypassed if temporary unavailable and the private database is used very little.

Usability: Expert users can access the system's functionalities on the back-end, using generic Ethereum and IPFS tools; in this way they can check the correctness of involved processes directly on the public ledger. On the other hand, non-expert users can still access these functionalities by means of the web-based front-end; in this case no knowledge about public ledgers, Ethereum and consensus protocols is needed. Moreover, the front-end is detached from the back-end, making thus easy to change it according to user's needs and preferences. In fact, it would be easy to add a mobile version of the front-end, in the form of a mobile app.

Law obligations: Circulation of artworks of high historical and cultural value is strictly regulated by legislation; this can be verified and enforced by the Government using the information provided by smart contracts and the administrator. Resale rights are paid automatically to the author in case of public sales (and checkable with administrator for private ones). Finally, the transactions on Ethereum (combined with IPFS) contain all the data requested for sales certificates. At the moment these transactions are not legally recognized yet but, in the meanwhile, they constitute a useful basis for producing the necessary documentation.

Cost analysis Transactions cost is an important parameter in defining the effective usability of the system. Every operation requires to pay a certain amount of gas in order to be performed. Gas is then converted in Ethereum based on a *gasPrice* specified by the sender. The higher the gasPrice is, the lower is the time expected for the transaction to be executed.

Tests have been done on Ropsten test network (among all the testnets, Ropsten is the most similar to the mainnet because it uses the same proof-of-work algorithm, but ethers stored in it have got no value). Setting the average gasPrice we can obtain a similar acceptance time to the one we would obtain in Ethereum if we set its average price (both prices are available in real time: at the time of tests, 1 Gwei on Ropsten was equivalent to 21 Gwei on the mainnet). The tests show that response time varies widely also for the same operation (from 10 seconds to 5 minutes). However, the waiting time was very rarely greater than 2 minutes. Furthermore, no significant correlation was found between the gas used and the waiting time, although results are inherent to the execution of functions. The deploy of smart contracts is also a transaction but has higher average gasPrice and acceptance time.

To determine the actual cost we used the estimations provided at `https://ethgasstation.info/`. At the time of test, the exchange was 1 ETH = 186.82 €, and the estimations were 10 Gwei for a average time of 3 hours, 21 Gwei for a average time of 2 minutes, and 32 Gwei for a average time of 30 seconds. The results from these tests are reported in Table 1. We can see that there are not prohibitive commissions for any of the offered operations. It is also important to note that tests have been carried out in a period of high commissions, and actual estimates for average acceptance time are much lower; moreover, the user is free to choose the price he prefers in relation to his time necessities and can also send multiple transactions concurrently (if they are not related). Beside these operations, the only expensive step is the Deployment itself, requiring about € 100 for a average acceptance time of 4 hours, but this operation is executed only once for the system startup and is paid by the administrator.

Of course, besides transaction costs there will be also fees for the administrator and authenticators for their work, but their definition goes beyond the scope of the current platform; e.g., they can be paid for each transaction, for each registration, *una tantum*, etc.

Operation	**Gas used**	**Cost (€)** slow	average	fast
addArtist()	165,499	0.309	0.649	0.989
addAuthenticator()	120,030	0.224	0.471	0.718
addBroker()	124,776	0.233	0.490	0.746
addInvestor()	97,308	0.182	0.382	0.582
registerArtwork()	313,946	0.587	1.232	1.877
startOpenSale()	101,621	0.190	0.399	0.608
startClosedSale()	134,853	0.252	0.529	0.806
buy()	133,093	0.249	0.522	0.796
buy() - *with broker*	176,391	0.330	0.692	1.055
sellPrivately()	107,559	0.201	0.422	0.643
entrustArtwork()	113,967	0.213	0.447	0.681
askForAuthentication()	126,949	0.237	0.498	0.759
addProposal()	228,131	0.426	0.895	1.364
authenticate()	74,602	0.139	0.293	0.446
registerAuthenticatedArtwork()	305,855	0.571	1.200	1.828
withdraw()	29,238	0.055	0.115	0.175

Table 1: Cost comparison for slow (gasPrice=10Gwei), average (gasPrice=21Gwei) and fast transactions (gasPrice=32Gwei).

5 Conclusions

In this paper we have proposed a solution for securing transactions of the art market, using Distributed Ledger Technology, and more precisely Ethereum's blockchain. This approach allows us to address the transparency, availability and integrity problems in a new and simpler way. The data and processing decentralization provided by Ethereum's DLT allows us to obtain good performances at a reasonable price, increasing considerably transparency (i.e., reducing at minimum the need to trust someone). In the art market, transparency is very important due to the presence of data of great interest to many people (i.e., all possible buyers). Higher transparency means more trust in the system, leading to greater willingness to invest.

The solution we have presented cannot be considered complete and definitive. A still open issue is how to match the exchange of the token with the exchange of the physical artwork: if a seller does not register a sale on the platform then the provenance chain breaks, and this could be interpreted as an illegal change of ownership (e.g., an illicit sale of a stolen artwork). Moreover, for distance sales, additional measures should be taken to ensure the real shipping of artworks. Solutions could include the use of a third party shipping company, where artworks should be deposited before the sale, or in obtaining the legal recognition of transactions on Ethereum. In fact, a legal endorsement would increase the attractiveness of the proposed solution.

A general issue concerns the actual *tokenization* of the artwork. In principle, we would like to associate the token to properties (e.g., physical-chemical) that uniquely identify the artwork in its state at the moment of the authentication; in this way, any alteration, e.g. due to negligent preservation, would invalidate the association betweem the artwork and the token therein. However it is very difficult to define these unalterable properties in general, as most of them can legitimately change during the time, e.g., for natural degradation of materials, for restorations, or even according to the will of the artist.

Another important issue concerns possibly wrong authentications of artworks. At the moment, these are invalidated (but not cancelled) by the current owner, so that a new one can be emitted. Another approach, more in the spirit of distributed ledgers, could be to have the authenticators to reach an agreement (possibly weighted by their respective reputation) over the token to be invalidated, without the intervention of an administrator.

Other interesting future developments can come from the integration of the blockchain with IoT devices. For example, we can keep the artwork in a sealed frame that can be unlocked (for a limited time) only with the private key of the current owner registered on blockchain. This would force every seller to register the sale in order to allow the buyer to examine the artwork. It also would be interesting to consider other blockchains; Ethereum itself is changing, with the plan of replacing the proof-of-work consensus algorithm with the cheaper and faster proof-of-stake algorithm. This improvement would increase the adoption of distributed ledger technologies for the traceability of values and properties, and in similar fields.

Finally, it would be interesting to provide a *formal* model of transactions over public ledgers. To this end, we consider to use the general framework of *bigraphic reactive systems* [11], which have been already proved to be effective in the context of multi-agent distributed systems [13].

Acknowledgments We thank HTS SrL for their help on technical and legislative aspects.

References

[1] Attuazione della direttiva 2001/84/CE, relativa al diritto dell'autore di un'opera d'arte sulle successive vendite dell'originale, d.Lgs. 13 febbraio 2006, n. 118

[2] Codice dei beni culturali e del paesaggio, d.Lgs. 22 gennaio 2004, n. 42

[3] Norme penali sulla contraffazione od alterazione di opere d'arte, l. 20 novembre 1971, n. 1062

[4] Regolamento per la semplificazione dei procedimenti. . . , d.P.R. 28 maggio 2001, n. 311

[5] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the 13th EuroSys Conference. p. 30. ACM (2018)

[6] The Art Basel and UBS Global Art Market Report 2019 (2019), https://www.artbasel.com/stories/art-market-report

[7] Benet, J.: Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561 (2014)

[8] Buterin, V.: A next-generation smart contract and decentralized application platform. white paper **3**, 37 (2014)

[9] Dannen, C.: Introducing Ethereum and Solidity. Springer (2017)

[10] Entriken, W., Shirley, D., Evans, J., Sachs, N.: Erc-721 non-fungible token standard. Ethereum Foundation (2018), https://eips.ethereum.org/EIPS/eip-721

[11] Grohmann, D., Miculan, M.: Reactive systems over directed bigraphs. In: Proc. CONCUR. Lecture Notes in Computer Science, vol. 4703, pp. 380–394. Springer (2007)

[12] Lee, W.M.: Using the MetaMask Chrome extension. In: Beginning Ethereum Smart Contracts Programming, pp. 93–126. Apress (2019)

[13] Mansutti, A., Miculan, M., Peressotti, M.: Multi-agent systems design and prototyping with bigraphical reactive systems. In: Proc. DAIS. Lecture Notes in Computer Science, vol. 8460, pp. 201–208. Springer (2014)

[14] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)

[15] Valenta, M., Sandner, P.: Comparison of Ethereum, Hyperledger Fabric and Corda. FSBC Working Paper (Jun 2017)

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