Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

SPECIAL ISSUE PAPER

BArt: Trading digital contents through digital assets

Cristina Pérez-Solà*[1,3] | Jordi Herrera-Joancomartí[2,3]

[1]Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya, Catalonia, Spain
[2]Dept. d'Enginyeria de la Informació i de les Comunicacions, Universitat Autònoma de Barcelona, Catalonia, Spain
[3]CYBERCAT, Centre de recerca en ciberseguretat de Catalunya, Catalonia, Spain

Correspondence
*Cristina Pérez-Solà, Rambla del Poblenou 156, Barcelona Email: cperezsola@uoc.edu

Summary

Since digital artworks are indeed digital content, they face the inherent problem digital content has: the link between content and its original author is very difficult to keep. Additionally, retaining control over digital copies of the content is also a challenging task. Digital coins have solved this very same problem through cryptocurrencies (for instance, Bitcoin) and have opened the door to apply the same techniques to other similar scenarios where ownership of digital assets needs to be preserved. In this paper, we propose BArt, a transparent and distributed mechanism for artists to commercialize their digital artwork, keeping control of the copies, monetizing its usage, and maintaining ownership. BArt allows artists to publicly register their work in the Bitcoin blockchain and sell usage rights in exchange for bitcoins. Buyers are allowed to exert the acquired rights. Proper behaviour from all parties is enforced by the system with cryptography and economic incentives.

KEYWORDS:
Digital assets, Blockchain technology, Bitcoin, Colored Coins.

1 | INTRODUCTION

One of the main challenges digital currencies face is to ensure the effective transfer of a coin when a payment is performed. Giving the digital nature of these systems and, as a consequence, the easiness of producing exact copies of content, preventing double-spending (the transference or usage of the same coin more than once) is the cornerstone of such schemes. Bitcoin solves this problem by using an immutable ledger in which all transactions are included. The main property of such ledger, known as the blockchain, is that it can be extended to add new information, but once any information is included, no modification is allowed. The interesting idea behind Bitcoin is that the immutable ledger is maintained in a distributed way. The special entities in the system that maintain the ledger are called miners, and their main task is to make the ledger grow by including new Bitcoin transactions into the blockchain. Information (i.e. Bitcoin transactions) inside the ledger is public and determines the ownership of every bitcoin in circulation. This mechanism was created (and used for the first time) by the Bitcoin cryptocurrency, and was afterwards popularized under the term blockchain technology. By providing this mechanism, Bitcoin opened the door to deal with similar environments where ownership of digital assets needs to be determinable at any time and transferable between different parties, for instance, shares of a company or, as we will discuss, digital artworks.

Indeed, the problem artists face when they try to commercialize their digital works is somehow similar to the one of digital currencies: how is it possible to prove ownership of a digital work? The artist, as the main author of the artwork, should be able to retain its authorship and, at the same time, he should be able to commercialize it, granting usage rights over the artwork and obtaining revenue in exchange for those rights. To that end, in a similar way than Bitcoin users prove that they own some bitcoins, buyers of digital content should be able to prove they are the legitimate owners of a digital work, or even that they possess some rights over the digital content (that have been granted by the legitimate owner of the work, usually in exchange of some amount of money).

In this paper we present `BArt`, a scheme that represents usage rights over digital artworks as digital assets in the Bitcoin blockchain. `BArt` allows artists to:

1. publicly register their work in the Bitcoin blockchain, allowing thereafter to **prove its ownership** in a transparent way;
2. **represent usage rights** over their work and **enforce** those rights through cryptography;
3. and **trade** those rights on top of Bitcoin (e.g. sell visualization rights over their works in exchange for bitcoins).

Since `BArt` is built on top of the Bitcoin blockchain, it has all the benefits that a distributed and public ledger has: **transparency** (all transactions involved in the system are public), **traceability** (artists can keep track of their artworks), **security** (it is possible for an owner of an asset to cryptographically prove ownership and, at the same time, it is not possible for an adversary to prove it), **decentralization** (there is no central authority in charge of the system), and **censorship-resistant** (there is no way to censor or forbid transactions between artists and buyers). Therefore, such approach serves both established artists and (new) emerging artists to commercialize their artwork, since it allows to perform all the procedure without the need of any large third-party content provider that manages the distribution and commercialization of the artworks.

The rest of the paper is organized as follows. Section 2 reviews some existing proposals that use blockchain technology for digital rights management. Section 3 provides an overview of the colored coin technology, a mechanism to define digital assets on top of the Bitcoin protocol. In Section 4, a general overview of `BArt` is provided, and its detailed description is given in Section 5. The main security and privacy properties of `BArt` are analyzed in Section 6. Finally, Section 7 concludes the paper and gives some guidelines for further research.

2 | RELATED WORK

Ascribe[1,2] is a platform that allows artists to create links between themselves and their works through the Bitcoin blockchain. It supports to transfer ownership, consign (grant the rights to transfer content to another user) or loan (display or use the content for a period of time) digital creations. Ascribe uses the SPOOL (Secure Public Online Ownership Ledger) blockchain protocol[3] to encode the needed information into the blockchain, i.e., Ascribe is not build on top of a generic colored coins protocol but uses its own protocol, designed specifically for managing creative works.

Mediachain[4] is a proposal for a protocol to associate metadata to creative works by using **a** blockchain to timestamp metadata, which is then stored in the InterPlanetary File System (IPFS). One of the differentiating features Mediachain offers is the usage of perceptual fingerprinting to find and aggregate information about different versions of the same image. The Mediachain Attribution Engine[5] is a project build on top of mediachain which has linked 125 million images from existing sharing platforms with their creators. It is now possible to search for those images using keywords, obtaining both the images and the information about their creators. Thanks to the perceptual fingerprinting engine, it is also possible to upload an image whose author is unknown to try to discover authorship information. If the exact same image is not found, the engine returns images and authorship information about similar visual works.

Both works are aimed to solve just a part of the problem artists face when distributing their creations, namely, to link artists with their works and/or to represent the rights granted over these works. With the proposed systems, one can securely link identities with works and transfer ownership of these works, but none of the reviewed systems provide means of **enforcing** the rights represented by the protocol or **monetizing** the creative work.

From the academic arena, there exists a report from the Blockchain For Creative Industries Research Cluster[6] that identifies the potential uses of blockchain technologies with respect to recorded pieces of music. The report is introductory and does not intend to provide specific solutions to the identified uses. Other academic papers that try to describe schemes for artistic content management on the blockchain just provide a general idea but lack any specific descriptions of the proposed systems[7,8,9].

In the same line of work, the JPEG committee has recently started a working group[10] for exploring the usage of blockchain technologies in their standard. They have published a report[11] that identifies the challenges the media industry faces where blockchain technologies can be of help, although no specific solutions are designed yet.

3 | BACKGROUND: REPRESENTING ASSETS IN THE BITCOIN BLOCKCHAIN

Bitcoin is a cryptocurrency based on public key cryptography. Bitcoin users have a *wallet* with a set of ECDSA key pairs: Bitcoin *addresses* are derived from public keys and are used as pseudonyms to which Bitcoins can be sent, whereas private keys are used to authorize payments through *signatures*. Payments are performed by broadcasting *transactions*, a data structure that specifies the movement of Bitcoins from a set of *inputs* to a set of *outputs*. Interested readers can refer to Narayanan et al. book[12] and Antonopoulos[13] for a complete explanation of Bitcoin.

Although Bitcoin is *only* a cryptocurrency by design, it's scripting language provides some flexibility. On the one hand, this flexibility allows to create transactions that can be redeemed with special scripts, other than just a user signing to release funds. This allows to build complex protocols on top of Bitcoin, that accomplish much more than just transferring bitcoin funds from one user to another user. On the other hand, Bitcoin's scripting language is also capable of storing some bytes of additional information in Bitcoin transactions, in such a way that transactions are still valid and can be redeemed. These additional bytes can be used to encode all sorts of metadata in standard Bitcoin transactions.

The concept of colored coins is based on the idea of representing real world assets on top of the existing Bitcoin system[†]. With colored coins, one is able to create a representation of an asset in the Bitcoin blockchain and manage this asset, for instance, transferring ownership or adding metadata with information about the asset.

An entity that creates an asset in the blockchain is called the asset issuer. Asset issuers compromise themselves to exchange the digital tokens they issue by the real world asset they represent, hence allowing the digital tokens to have value. An entity in possession of asset tokens may then redeem the tokens by the real world assets they represent.

Colored coins may be used, among other use cases, to issue financial assets (like stocks or bonds[14,15]), to certify credentials (such as academic certificates[16]), or to prove existence of digital documents (e.g, with a timestamping service[17]).

Because colored coins are built on top of Bitcoin, they also share it's limitations. That is, the scalability of a system using colored coins is limited by the scalability offered by Bitcoin. Moreover, Bitcoin transaction fees also apply to transactions carrying colored coins metadata. However, solutions like the Lightning Network, a second layer payment protocol that overcomes these limitations, can also be integrated with colored coins, and thus be used to go around the abovementioned limitations.

Currently, there exist several protocol proposals[18,19,20,21], at different maturation stages, that instantiate the general idea of colored coins. Each of these protocols may have one or many actual implementations of the protocol. The exact way in which the Bitcoin scripting language is used to carry information about assets is defined by each individual colored coins protocol. In a similar way, asset manipulation rules are defined individually by each protocol and thus may differ from one protocol to another. The set of available manipulation actions depends also on the specific protocol used.[‡]

3.1 | Colored Coins protocol

Colu's Colored Coins protocol[18] (from now on, we will be using the expression colored coins to refer to this specific proposal) is one of the proposals that offers the most functionalities, even though some of them are documented but still not implemented. They currently have a working API that allows to issue and transfer assets. Apart from the API, they also offer some graphical interfaces that allow non technical users to perform basic operations within the system.

Colored coins transfer and issuance manipulation instructions are stored in an `OP_RETURN` output. Each Bitcoin transaction may only have at most one `OP_RETURN` output. This output is used to encode the basic properties of the issued asset (on issuance) or the transfer instructions (on transference).

Additionally, metadata may be included into the colored transactions. Metadata is stored in torrents and a hash of the metadata is included in the transaction, thus assuring integrity. In order to include this hash, additional space may be needed in the transaction to encode it. If it is the case, colored coins use 1-of-2 and 1-of-3 multisignature addresses to obtain this additional space.

[†]Although theoretically, we could create digital assets on top of other Blockchains, in this paper we refer to the most extended definition of colored coins, that includes assets represented on top of Bitcoin.

[‡]Bitcoin is not the only permissionless cryptocurrency that is able to represent and trade digital assets. Ethereum, the second largest cryptocurrency in terms of market capitalization, is able to handle digital assets natively. Since `BArt` needs to associate unique information to each token and tokens are not equivalent, implementing `BArt` over Ethereum would require to create non-fungible tokens, whose issuance and transference will be controlled by a set of smart contracts. Therefore, using Ethereum instead of Bitcoin colored-coins could have been an alternative to the proposed solution. Using permissioned blockchains may also be an alternative to create a system with similar properties, although in this case the overall approach of the scheme and the problems to solve will be different, since the blockchain infrastructure would have to be designed and created, and incentives for all the parties would have to be adjusted.

The two operations that can be made with colored coins assets are asset **issuance** and asset **transfer**.

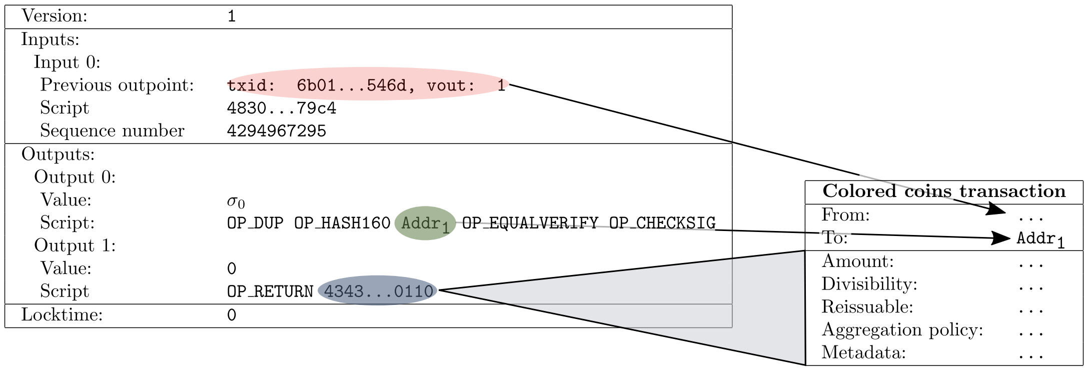

FIGURE 1 Example of a Bitcoin transaction with colored coins data embedded into an `OP_RETURN` output.

Figure 1 shows an example of a Bitcoin transaction with colored coins data embedded into an `OP_RETURN` output. The Bitcoin transaction has only one input, and thus the colored coins transaction will also have the same input. If the colored coins transaction is a transfer transaction, the assets on that input will be moved, following the instructions encoded in the transaction. On the contrary, the Bitcoin transaction has two outputs: one of them (in the example, the first one) will be interpreted also as an output of the colored coins protocol (and thus assets can be moved to the address specified in the output), whereas the other one contains an `OP_RETURN` opcode that encodes colored coins issuance or transfer instructions. In the general case, Bitcoin transactions encoding colored coins data will always have one single `OP_RETURN` output encoding instructions for moving or issuing assets to the rest of the outputs.

For simplicity, in the rest of the paper we specify only the contents of the colored coins transactions, without detailing the specific encoding of the colored coin transaction inside a Bitcoin transaction.

3.1.1 | Asset issuance

Asset issuance allows to create new assets. Some of the properties of the created asset are assigned at the creation time:

- The **amount** of issued assets (the number of assets issued).
- The **divisibility** of the asset, which allows to specify the smallest tradeable unit of the asset. Some assets will be indivisible (e.g., assets representing tickets for a concert) while others may allow some kind of partitioning (e.g., a currency may allow to exchange half of a unit).
- The ability to **reissue** more units of the same asset. When an asset is issued, the issuer may decide to lock it from further issuances. In that case, no one (not even the issuer himself) can make any subsequent issuance of the same asset. On the contrary, assets can be issued in an unlocked fashion, allowing the issuer to create more units of the same asset in the future.
- The **aggregation policy** that specifies whether different units of the same asset are the same. Some assets will be aggregatable (e.g. cinema tickets with unnumbered seats), such that they can be transferred together, whereas other assets will be dispersed (e.g. cinema tickets with a single seat assigned), needing individual treatment.
- Asset **metadata** with additional information associated to the asset. The protocol describes two classes of metadata: **static data** and **rules**. In turn, static data includes both protocol defined data (that is used by the colored coins protocol) and user data (that can be arbitrarily added by issuers to convey additional information from the asset). Rules are used to further restrict how assets can be traded or created.

Additionally, when an asset is issued it obtains an identifier, i.e., an immutable string that identifies the issued asset. The asset identifier is derived from the reissuance, divisibility and aggregation policies of the asset, together with some information from

Issuance transaction	
From:	$\texttt{UTXO}_n$
To:	$\texttt{Addr}_1$
Amount:	200
Divisibility:	0 (no decimals allowed)
Reissuable:	True
Aggregation policy:	Aggregatable
Metadata:	
Static data:	
Asset name:	Gala dinner ticket
Issuer:	International Association Of Gala Dinners
Description:	This asset represents tickets for the gala dinner
Verifications:	https://iagd.com/assetfile
User data:	{place: a very nice hotel}
Rules:	
Fees:	No fees - inheritance 0 (no new fees may be added)
Expiration:	Expires on 08-04-2019 - inheritance 0 (no changes can be made)

TABLE 1 Example of an issuance transaction.

the unspent transaction output (UTXO) referenced in the first input of the issuance transaction.[§] However, note that the asset identifier is not chosen by the issuer, since it is derived directly from properties the issuer does not fully control.

Asset metadata provides additional functionalities that are needed in order to deploy a secure colored coins system. In the next paragraphs, we review the functionalities that are used, afterwards, in our proposed scheme.

Static metadata includes, among others, **issuer verification** data. In order to allow the colored coins assets to have value, users of the system need to trust their issuer, because they need to be sure that the digital tokens can be exchanged by the asset they represent. One of the ways to obtain this trust is to verify the issuer, for instance, by linking the issuer of an asset to an existing company or individual the users already trust. One of the alternatives the colored coins protocol provides to achieve this functionality is by linking the issuer to a website protected by https: a url of the company website behind https is included in the metadata of the issuance transaction and a text file with the asset ids is created in the given url.

Rules allow to specify additional constraints about how the asset may be issued and transferred. Our proposal uses two different kinds of rules: **fees** and **expiration**. Assets may have a fee, that is, a price to pay for asset transference. In order to specify a rule concerning fees, one must declare the asset in which the fees are payed (it may be BTC or any colored coins asset), the amount, and the address to which the fee must be paid. The expiration rule allows to create assets that have a caducity. Expiration can be specified either with a given block height or with number of blocks mined on top of the issuance transaction.

Rules in the colored coins protocol have also an **inheritance** parameter, that allows to specify how are the rules transferred when the asset is transferred. For instance, different inheritance levels in fees rules allows to specify whether the recipient may add additional fees to the transferred asset.

Table 1 provides an example of data encoded in an issuance transaction. For the sake of simplicity, we have omitted some of the fields that the protocol allows to include but that are not used in our proposal.

[§]The exact information from the issuance transaction that is used for creating the identifier depends on the reissuance policy of the asset. Unlocked asset ids consider the output script of the UTXO referenced in the first input of the issuance transaction. Instead, locked asset ids are created from the transaction identifier and the index of the Bitcoin UTXO referenced in the first input of the issuance transaction. In this way, the reissuance policy is embedded into the assets identifier, since it won't be possible to create more assets with the same id for locked assets, while it is indeed possible for unlocked assets.

3.1.2 | Asset transfer

Assets can be transferred between Bitcoin addresses. The colored coins protocol allows to trade multiple assets in a single transaction. For instance, one colored coins transaction may have a single input with ten assets of type 1 and five assets of type 2, and may distribute those assets between five outputs, assigning two assets of type 1 and one asset of type 2 to each of the outputs. In this way, we can say that the colored coins protocol colors UTXOs (because it assigns assets to UTXOs), so that whoever is able to spend that UTXO becomes the owner of the asset.

Transfer transaction		
From:	$\texttt{UTXO}_1$	
To:	$\texttt{Addr}_{\texttt{attendant1}}$	2
	$\texttt{Addr}_{\texttt{attendant2}}$	1
	$\texttt{Addr}_{\texttt{attendant3}}$	20
	$\texttt{Addr}_1$	177
Metadata:		
Static data:		
Asset id:	`Ua3WSEuChgqwsip9nHkmCrz4XELfkn4agKkvNN`	
Asset name:	`Gala dinner ticket`	
Issuer:	`International Association Of Gala Dinners`	
Description:	`This asset represents tickets for the dinner`	
Verifications:	`https://iagd.com/assetfile`	
User data:	`{place: a very nice hotel}`	
Rules:		
Fees:	`No fees`	
Expiration:	`Expires on 08-04-2019`	

TABLE 2 Example of a transfer transaction.

Colored coins transfer transactions may also include metadata. For instance, they may include the asset id to ease processing or information about the physical asset they represent.

Note that asset transference must follow the rules specified during the issuance procedure.

Table 2 provides an example of data encoded in a transfer transaction.

4 | OVERVIEW OF `BArt`

This section provides a high-level description of `BArt`, presenting the participating actors, the actions they can perform, the usage of colored-coins assets in the system, and the desired security properties. Then, Section 5 provides a detailed description of each of the procedures in the protocol, together with the colored coins transactions involved in each of them.

Three different types of actors participate into `BArt`. An **Artist** `A` (that may be an artist himself, a group of artists cooperating together or even a producer) has some content c (any digital media, for instance, a song, a movie, a document, etc.) he wants to distribute. A **Buyer** `B` is interested in acquiring some rights for the content c. The buyer is in possession of a Player `P`, that is able to reproduce or allow the Buyer to visualize the content. The player (a software or hardware device) has been developed by a certain **Player Manufacturer** `PM`, that is incentivized to well-behave within the system. Figure 2 summarizes the actions each kind of actor may perform within the system and the entities affected by those actions.

Each of the actors must perform an initialization procedure the first time they join the system (the initialization phase for `PM`, `A`, and `B` is described, respectively, in Sections 5.1.1, 5.1.2, and 5.1.3). Then, once the actors are initialized, each of them may perform a different set of actions. The player manufacturer `PM` is able to create (and authorize) new players (Section 5.2.1) or revoke the authorizations he has conceded to players that misbehave (Section 5.2.2). The artist `A` may add new content to the system (Section 5.3.1). Finally, both the artist `A` and the buyer `B` may cooperate so that the buyer ends up reproducing a content

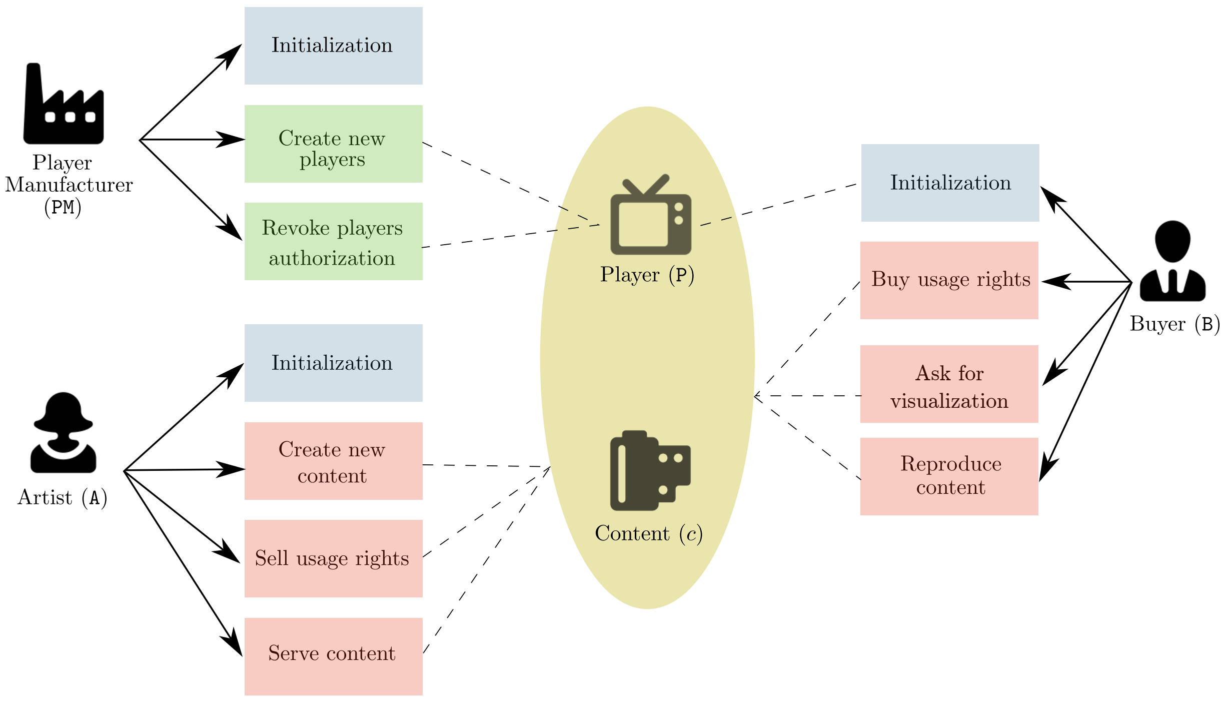

FIGURE 2 Overview of the entities in `BArt`. Actions are colored by purpose: inicialization actions are colored in blue, player authorization actions in green, and content-related actions in red.

created by the artist: first, `B` buys some usage rights for a content c that `A` has created (Section 5.3.2); then, `A` sells the requested usage rights (Section 5.3.3); afterwards `B` may want to make effective the usage of the rights he has bough (Section 5.3.4); in response `A` gives him the information he needs to reproduce the content (Section 5.3.5); and finally `B` reproduces the content (Section 5.3.6).

`BArt` uses colored coins assets to create a system where artists can claim authorship of their works, represent and enforce usage rights for those works, and trade them in exchange for bitcoins. In particular, `BArt` uses two different kinds of assets: Player Authorization assets (*AssetPlayerAuth*) and Visualization Rights Assets (*AssetVisRights*). Player authorization assets are created by `PM` during his initialization and transferred afterwards to their players `P` in order to certify them. Thus certified players will be in possession of authorization assets, which will allow them to play or visualize the contents. If any of the players misbehaves, its `PM` may revoke its authorization rights. The visualization asset is issued by `A` whenever new content c is created. It describes the desired policy for the visualization of the content, that includes the usage rights of such content, for instance, expiration date, valid time frame, number of times the content can be used, etc. `A` will trade its contents by selling to `B` the corresponding asset that would enable `B` both to prove the possession of the designated rights and the secure access to the content. Table 3 summarizes the kinds of assets created in our system.

Asset	Represents	Created by	Transferred to
AssetVisRights	Usage rights for a content c.	`A`	`B` & `P`
AssetPlayerAuth	Authorization from a `PM`.	`PM`	`P`

TABLE 3 Assets summary.

In `BArt`, `B` will be able to reproduce a content c in a player `P` only when:

- `P` has been authorized by `PM` through an authorization asset,
- `A` recognizes `PM` as a valid producer for the given content (as detailed on the policy of the content c asset), and

- B has obtained a visualization asset from A.

The reproduction of a particular content c provides revenue both to A and to PM. The payment to A will be made through the purchase of a visualization asset while the payment to PM will be performed directly when the content is reproduced.

Table 4 summarizes the notation used to describe the scheme.

Entities	
A	Artist
B	Buyer
PM	Player manufacturer
P	Player
c	Content
Keys	
$(\mathrm{SK}, \mathrm{PK})$	Private and public key pair
$\mathrm{Addr}[\mathrm{PK}_\mathrm{U}]$	Bitcoin address from user U associated to a public key
k	Symmetric key
HDW_U	Hierarchical Deterministic Wallet from user U
Encryption	
$\overline{c} = E_k(c)$	Encrypted content
$\hat{c} = E_{PK_\mathrm{P}}(k)$	Encrypted k (which decrypts $\overline{c}$)
Assets	
AssetPlayerAuth	Asset representing player authorization rights
AssetVisRights	Asset representing visualization rights

TABLE 4 Notation summary.

BArt allows both to represent usage rights for a specific content and to enforce those rights. With respect to the enforcement of rights, the security goals our system aims to achieve are:

- Buyers should not be able to reproduce a content without first buying the adequate rights. Buyers can just reproduce the content adhering to the rights they have bought, and need to have an authorized player in order to reproduce content.
- Malicious behaviour from artists should be detectable. If artists misbehave, affected users should be able to create proofs of the fraud verifiable by any external observer (with access to publicly available information stored in the Bitcoin blockchain).
- Malicious behaviour from players and player manufacturers should be desincentivized (the system must give incentives to player manufacturers to behave as expected).

5 | DETAILED DESCRIPTION OF BArt

We have provided a high level overview of the proposed system. In this section, we describe in detail each of the procedures that can happen within BArt, together with the colored coins transactions involved in each of them. The procedures are grouped by purpose: Section 5.1 details the initialization procedures for each of the three types of actors in the system (detailed in blue in Figure 2); Section 5.2 describes the actions involving players' authorization (in green in Fig. 2); and Section 5.3 explains the procedures affecting content (red actions in Fig. 2). Finally, Section 5.4 presents a summary of all the Bitcoin transactions that may appear in the system.

5.1 | Initialization

The first time a new actor joins `BArt`, he must perform an initialization step. The exact set of actions conforming this initialization depends on the type of actor.

5.1.1 | A new Player Manufacturer joins the system

The first time a Player Manufacturer (`PM`) operates within the system, he performs the initialization phase consisting on two different tasks:

1. Create a Bitcoin wallet.
2. Issue an asset representing authorization rights for their players (*AssetPlayerAuth*).

Concerning wallet creation, `PM` needs to have a hierarchical deterministic wallet[¶]. Therefore, `PM` generates a new master seed and creates a wallet from the seed. Once the wallet, $\mathrm{HDW}_{\mathtt{PM}}$, has been created, `PM` obtains an address from the wallet, $\mathrm{FeeAddr}_{\mathtt{PM}}$= $\mathrm{Addr}(\mathrm{PK}^0_{\mathtt{PM}})$, that will be used for issuing the asset and for collecting the fees `PM` will receive for each reproduction of content any of his players performs.

Afterwards, `PM` issues the asset Player Authorization Rights (*AssetPlayerAuth*). Units of this asset will be transferred to players manufactured by `PM` in order to allow them to prove they are authorized by the `PM`.

***AssetPlayerAuth* Issuance transaction**	
From:	$\mathrm{FeeAddr}_{\mathtt{PM}}$
To:	$\mathrm{FeeAddr}_{\mathtt{PM}}$
Amount:	n_{auth}
Divisibility:	`0` (no decimals allowed)
Reissuable:	`True`
Aggregation policy:	`Dispersed`
Metadata:	
Static data:	
Asset name:	`Player Authorization Rights PM`
Issuer:	`PM`
Description:	`This asset represents player authorization rights from PM`
Verifications:	`https://PM/assetfile`
User data:	`{price per vis.:` σ_{pm} `BTC}`
Rules:	
Fees:	`No fees - inheritance 1` (the recipient may add fees)
Expiration:	`No expiration - inheritance 0` (no changes can be made)

TABLE 5 Tx_0: *AssetPlayerAuth* Issuance transaction.

AssetPlayerAuth is issued by sending a Bitcoin transaction,Tx_0, with colored coins data to the Bitcoin network. Table 5 describes the colored coins information encoded in Tx_0. Since `PM` may not know in advance the exact number of units of the asset needed, the asset will be unlocked, and thus the asset can be reissued in the future if more units are needed. The minimum unit of *AssetPlayerAuth* will be 1 (hence divisibility is set to 0, allowing only integer divisions of units). Furthermore, the asset will be dispersed, so that each unit of the asset is treated by separate transfer instructions and maintains its own metadata. Regarding rules, no fees nor expiration will be set. However, fees will be open, so that the recipient of the asset can add fees to further transactions involving the asset. Additionally, the player manufacturer informs about the price per visualization he sets

[¶]See Hierarchical Deterministic Wallets informational BIP 32[22] for a detailed description of the properties and implementation details of HDW.

for players authorized by this asset, σ_{pm}. This price will be payed for each reproduction of content in the system to PM, and acts as incentive for players to well-behave. Note that including σ_{pm} in the metadata has only informative purposes: in the procedure detailed afterwards in Section 5.2.1 this price is added as a fee, and thus it's payment is enforced by the colored coins protocol.

A PM needs to perform the Player Manufacturer setup phase only once (when PM joins the system). However, if PM transfers all units of *AssetPlayerAuth* to its players and needs to create more units of the asset, PM may send another transaction to the Bitcoin network reissuing more units of *AssetPlayerAuth*. Note that, in this case, there is no need to generate a new wallet (the same address, $\text{FeeAddr}_{\texttt{PM}}$, is used for the reissuance).

5.1.2 | A new Artist joins the system

The first time an artist joins BArt, he proceeds to perform the initialization phase:

1. Create a Bitcoin wallet.

In this case, the wallet does not necessarily have to be a HDW. Addresses from this wallet will be used to issue assets representing usage rights for the artist's content. Note that there is no need for the artist to communicate with any other party within the system at this moment (wallet creation may be done offline).

An artist only needs to create a wallet once, the first time he joins the system.

5.1.3 | A new Buyer joins the system

Similarly, the first time a buyer joins BArt, he initializes by performing one task:

1. Create a Bitcoin wallet.

Again, the wallet does not necessarily have to be a HDW. Addresses from this wallet will be used to pay artists for content, to store visualization assets already bought, and to transfer those assets in exchange for the content. Again, there is no need for the buyer to communicate with any other party within the system at this moment.

A buyer only needs to create a wallet once, the first time he joins the system.

5.2 | Players' authorization

When new players are created, they are authorized by their player manufacturer, who may decide to revoke this authorization afterwards (if a player is found to be misbehaving). This section describes the procedures related to authorization and revocation of players.

5.2.1 | New Players are created

Once a Player Manufacturer has joined the system (Section 5.1.1), he may proceed to create and authorize its players P. This procedure is performed before the players are distributed to end users. The Player setup phase consists in two steps:

1. Create a Bitcoin wallet (derived from $\text{HDW}_{\texttt{PM}}$).
2. Authorize the player by transferring *AssetPlayerAuth* to addresses on the player's wallet.

With respect to wallet creation, in this case PM derives a new wallet account from the master wallet, $\text{HDW}_{\texttt{PM}}$, and saves it into P (specifically, the child private extended key is stored into P). By deriving a new wallet account from $\text{HDW}_{\texttt{PM}}$ and storing it in the player, the player manufacturer is able to maintain control over all addresses any of his players generate, seeing all transactions involving the player and, at the same time, being able to operate with them. This allows PM to revoke the authorization rights of any player if he misbehaves. Moreover, it allows to do so in an efficient way, since player addresses can be derived from information known by the PM and the PM does not need to store all players' private keys individually. Note that the usage of HDW reduces the amount of space the PM needs to use to store all the players private keys, simplifies key backup and recovery procedures, and allows to compartmentalize keys. As an alternative to using a HDW, the PM could use a non-HD wallet, and maintain a list with all the private keys of all players he have produced over time. However, this is costly in terms of storage and key management, and does not provide any native way to organize keys.

Transfer transaction		
From:	FeeAddr$_{\mathtt{PM}}$	
To:	Addr(PK$^1_{\mathtt{P}}$)	1 *AssetPlayerAuth*
	...	1 *AssetPlayerAuth*
	Addr(PK$^m_{\mathtt{P}}$)	1 *AssetPlayerAuth*
	...	...
Metadata:		
Static data:		
Asset name:	`Player Authorization Rights PM`	
Issuer:	`PM`	
Description:	`This asset represents player authorization rights from PM`	
Verifications:	`https://PM/assetfile`	
User data:	`{price per vis.:` σ_{pm} `BTC}`	
Rules:		
Fees:	`1 unit of` *AssetPlayerAuth* `to FeeAddr`$_{\mathtt{PM}}$ `- inheritance 0` (no new fees may be added)	
Expiration:	`No expiration - inheritance 0` (no changes can be made)	

TABLE 6 Tx$_1$: *AssetPlayerAuth* units are transferred to the player.

Regarding player authorization, PM generates and authorizes m addresses of P, Addr(PK$^j_{\mathtt{P}}$) for $j = 1, \cdots, m$, by transferring 1 unit of player authorization rights, *AssetPlayerAuth*, to each generated player address.

AssetPlayerAuth are transferred by sending a colored coins asset transfer transaction, Tx$_1$, to the Bitcoin network. Table 6 describes the contents of Tx$_1$. The transaction source address corresponds to FeeAddr$_{\mathtt{PM}}$, the address that has all the units of *AssetPlayerAuth* after issuance. Transaction outputs send 1 unit of the *AssetPlayerAuth* to each of the m addresses generated for the player. Note that the same transaction can be used to authorize different players. As we will see later in Section 6, using the same transaction for authorizing multiple players is recommended, since it improves privacy while minimizing costs.

PM needs to ensure that *AssetPlayerAuth* units are non-transferable once they have been transferred to a player. By doing so, the PM ensures that he retains the power to revoke players authorization rights if any player starts misbehaving. For this reason, a fee is added, such that any further transfer of the asset needs to pay 1 unit of the asset to FeeAddr$_{\mathtt{PM}}$.

A PM may add new players to the system whenever he wants.

5.2.2 | Player rights revocation

If a player P starts misbehaving, its PM can revoke his privileges:

1. Transfer all units of *AssetPlayerAuth* from the player back to the player manufacturer.

By transferring all units of *AssetPlayerAuth* from the player back to an address controlled by the PM alone (the address used to collect fees), PM is effectively revoking the authorization to the player. Because the wallet included in the player is a child account of a HDW whose master seed is known by the player manufacturer, PM can derive any of the players private keys and transfer any asset the player has to someone else by sending Tx$_2$ to the Bitcoin network (Table 7 shows the colored coins content of Tx$_2$).

5.3 | Content management

The main purpose of BArt is to be able to properly manage content, handling the usage rights the artist decides to grant and providing the means for the buyer to acquire, pay, and use these rights. This section describes the procedures related to content management within the system.

Transfer transaction		
From:	Addr($\mathrm{PK}^i_{\mathtt{P}}$)	
To:	FeeAddr$_{\mathtt{PM}}$	1 *AssetPlayerAuth*
Metadata:		
Static data:		
Asset name:	`Player Authorization Rights PM`	
Issuer:	`PM`	
Description:	`This asset represents player authorization rights from PM`	
Verifications:	`https://PM/assetfile`	
User data:	`{price per vis.:` σ_{pm} `BTC}`	
Rules:		
Fees:	`1 unit of` *AssetPlayerAuth* `to FeeAddr`$_{\mathtt{PM}}$ `- inheritance 0` (no new fees may be added)	
Expiration:	`No expiration - inheritance 0` (no changes can be made)	

TABLE 7 Tx_2: *AssetPlayerAuth* units are transferred back to the player manufacturer.

5.3.1 | New Content is created

Whenever an artist A wants to make his new content c available on the network, he proceeds to:

1. Freely distribute the encrypted content.
2. Issue an asset *AssetVisRights*, representing visualization rights of content c.

In order to create an encrypted version of the content c, A first generates a random symmetric key k. Then, he encrypts the content c with the symmetric key k and obtains $\overline{c} = E_k(c)$. Finally, A freely distributes the encrypted content $\overline{c}$. The specific technology used by the A to distribute the encrypted content is not defined by our protocol, that is, the A can chose how to distribute the encrypted content. The decision will be influenced by the size of the content, the number of downloads she expects to have, or the security properties she wants to guarantee. One of the possible alternatives is to upload the file to the Interplanetary File System (IPFS). Being a decentralized peer-to-peer protocol, allowing to handle high volumes of data (there is no limit on the size of the uploaded data) with high efficiency, and offering resilience, IPFS is a good complement to `BArt`.

With respect to asset issuance, A derives a new address from his wallet, Addr($\mathrm{PK}_{\mathtt{A}}$), and issues n_{vis} units of *AssetVisRights*, the asset representing the usage rights for content c, by sending the colored coins transaction Tx_3 to the Bitcoin network. Units of *AssetVisRights* will be bought afterwards by interested buyers in order to acquire the rights to visualize c together with the means to do so. Table 8 summarizes the colored coins information encoded in Tx_3.

Metadata of the asset includes the hash of the content, $H(c)$, the hash of the encrypted content, $H(\overline{c})$, and the hash of the decryption key $H(k)$. These values are used as commitments made by the artist, that is, by sending Tx_3 A is claiming that the k whose hash is $H(k)$ can be used to correctly decrypt the content $\overline{c}$ whose hash is $H(\overline{c})$, and that the result of the decryption will be the content c, with hash $H(c)$. As we will explain later in Section 6, those values can be used to demonstrate that a certain A is performing fraud (if this is, indeed, the case).

Note that the artist will use an independent address for each content he distributes. This allows users to ask for a specific content using the Bitcoin P2P network, without the need for any additional channels between A and B. Buyers may discover the address associated to a specific content by looking at the assets' metadata (which includes the name and the description of the content). A list of all assets from a given content producer can be found in the assets file under the A domain. Additionally, A may provide other interfaces for users to access this information, e.g. a content browser in their site.

An artist may add new content to the system as many times as he wishes. Each new content added to the system will have its own asset, representing its usage rights.

5.3.2 | Buyer acquires visualization rights

Whenever a user wants to acquire usage rights for a certain content c, he proceeds as follows:

***AssetVis* Issuance transaction**	
From:	Addr($PK_{\texttt{A}}$)
To:	Addr($PK_{\texttt{A}}$)
Amount:	n_{vis}
Divisibility:	`0` (no decimals allowed)
Reissuable:	`True`
Aggregation policy:	`Dispersed`
Metadata:	
Static data:	
Asset name:	`Content Authorization Rights for content` c
Issuer:	`A`
Description:	`A description of the content (for instance, a summary, size of the file, codification, etc.)`
Verifications:	`https://A/assetfile`
User data:	`{hash of the content:` $H(c)$`,` `hash of the encrypted content:` $H(\overline{c})$`,` `hash of the decryption key:` $H(k)$`,` `price per visualization:` σ_a `BTC,` `PM list: list of accepted PM with their fee collecting addresses}`
Rules:	
Fees:	`No fees - inheritance 1` (the recipient may add fees)
Expiration:	`No expiration - inheritance 0` (no changes can be made)

TABLE 8 Tx_3: *AssetVis* Issuance transaction.

1. Obtains metadata associated to content c
2. Pays for the usage rights of c.

A user interested in acquiring visualization rights for a given content c first retrieves the metadata associated to the visualization rights asset of content c. With the information contained in the asset's metadata, the user is able to check the identity of the issuer (by accessing the `verification` url and checking the validity of the SSL certificate) and to verify that the player manufacturer `PM` of the player `P` in his possession is recognized to reproduce the content (by looking at the `player manufacturer list` under the `user data` field). The buyer also learns the price per visualization the issuer has set for the content, σ_a, and the Bitcoin address used to issue the asset, Addr($PK_{\texttt{A}}$).

If all the validations are successful and the buyer wants to proceed with the purchase, the buyer sends a transaction Tx_4 to Addr($PK_{\texttt{A}}$) paying σ_b bitcoins. The input of the transaction has a particularity: it must spend a 2-out-of-2 multisignature output, with one public key belonging to the user's wallet and the other public key belonging to the player's `P` wallet. That is, the output Tx_4 spends is of the form: `2 <pubKeyPlayer> <pubKeyBuyer> 2 OP_CHECKMULTISIG`. The purpose of this transaction is threefold: 1) pays `A` for the visualization rights 2) asks `A` for visualization rights for content c and 3) informs `A` about the player's public key, $PK^j_{\texttt{P}}$. Note that Tx_4 is a normal Bitcoin transaction that does not contain any colored coins metadata appended to it (no assets are issued nor transferred by this transaction).

5.3.3 | Artist sells visualization rights

Upon reception of a transaction (Tx_4) paying bitcoins to any of the artist's addresses used to issue *AssetVisRights*, the artists proceeds to:

1. Validate the received transaction, Tx_4.
2. Transfer *AssetVisRights* to the payer.

First, with respect to transaction validation, note that `A` receives Tx_4 through the Bitcoin network, and thus the transaction will already be a valid Bitcoin transaction. `A` needs to check whether one of the public keys in the output that Tx_4 is spending has authorization from a manufacturer to operate. That is, the address has authorization assets (*AssetPlayerAuth*) from a player manufacturer recognized to operate with the requested content (as specified in the metadata of the issuance transaction of the asset, Tx_3). If it does, `A` retrieves the fee collecting address from the player's manufacturer (for instance, by looking at the input of the transaction that transferred the assets to the player's address, Tx_1).

Transfer transaction		
From:	Addr(PK$_\mathtt{A}$)	
To:	Addr(PK$_\mathtt{B}$, PK$^j_\mathtt{P}$)	σ_b/σ_a *AssetVisRights*
Metadata:		
Static data:		
Asset name:	`Content Authorization Rights for content` c	
Issuer:	`A`	
Description:	`A description of the content (for instance, a summary, size of the file, codification, etc.)`	
Verifications:	`https://A/assetfile`	
User data:	`{hash of the content:` $H(c)$`,` `hash of the encrypted content:` $H(\overline{c})$`,` `hash of the decryption key:` $H(k)$`,` `price per visualization:` σ_a `BTC,` `PM list: list of accepted PM with their fee collecting addresses}`	
Rules:		
Fees:	`1 unit of` *AssetVis* `to Addr(PK`$_\mathtt{A}$`)` σ_{pm} BTC `to FeeAddr`$_\mathtt{PM}$ `inheritance 0` (no changes can be made)	
Expiration:	`No expiration - inheritance 0` (no changes can be made)	

TABLE 9 Tx_5: $\frac{\sigma_b}{\sigma_a}$ units of *AssetVisRights* are transferred to the user and the player.

Then, `A` sends a transaction Tx_5 to the Bitcoin network transferring visualization rights for content c to both `B` and `P` by using a multisignature output (a 2-out-of-2 multisig script with a public key from the buyer and a public key from the player). Note that at this point, neither the user nor the player alone are able to redeem the visualization rights. The exact amount of visualization assets (*AssetVisRights*) is computed from the price of the asset σ_a (as displayed in the asset's metadata of Tx_3 under the `price per visualization` tag) and the amount of bitcoins payed to `A` in Tx_4, σ_b.

Table 9 presents the colored coins information encoded in Tx_5. In this transaction, `A` adds a set of rules regarding fees that ensure the correct operation of the system. On the one hand, each transfer of *AssetVisRights* assets needs to pay back 1 unit of *AssetVisRights* to `A`, thus effectively spending a visualization token. Moreover, each transfer also pays a fixed amount of bitcoins to the player manufacturer `PM`.

An artist will execute this procedure each time he receives bitcoins to any of the addresses he has used to issue visualization rights assets for any of his content.

5.3.4 | Buyer asks to visualize content

Once the buyer `B` has acquired visualization rights for a given content c, he may decide to use them to actually visualize the content. Then, `B` proceeds as follows:

1. Transfers 1 unit of *AssetVisRights* back to the artist (paying the corresponding fee to the player manufacturer).

Transfer transaction		
From:	Addr(PK_B, PK_P^j)	
To:	Addr(PK_A)	1 *AssetVis*
	FeeAddr$_{PM}$	σ_{pm} BTC
Metadata:		
Static data:		
Asset name:	`Content Authorization Rights for content` c	
Issuer:	`A`	
Description:	`A description of the content (for instance, a summary, size of the file, codification, etc.)`	
Verifications:	`https://A/assetfile`	
User data:	`{hash of the content:` $H(c)$`,` `hash of the encrypted content:` $H(\overline{c})$`,` `hash of the decryption key:` $H(k)$`,` `price per visualization:` σ_a `BTC,` `PM list: list of accepted PM with their fee collecting addresses}`	
Rules:		
Fees:	`1 unit of` *AssetVis* `to Addr(PK`$_A$`)` σ_{pm} BTC `to FeeAddr`$_{PM}$ `- inheritance 0` (no changes can be made)	
Expiration:	`No expiration - inheritance 0` (no changes can be made)	

TABLE 10 Tx_6: A user visualizes content c.

In order to do so, he creates and sends a new transaction Tx_6 as depicted in Table 10. Tx_6 pays 1 visualization asset to `A` and σ_{pm} bitcoins to `PM`, complying with the colored coins fees specified by Tx_5. By sending Tx_6, the buyer is asking for the visualization key for content c and, at the same time, he is both providing proof that he is spending a visualization asset *AssetVisRights* in his possession and that he is paying the player manufacturer the adequate amount of bitcoins.

Recall that *AssetVisRights* was transferred by `A` to a multisignature output controlled both by the buyer and the player (by Tx_5). Therefore, Tx_6 needs to be signed by both the buyer and the player. The procedure to obtain the signature from both parties is as follows:

1. `B` requests the player to visualize content c.
2. `P` checks whether there are any visualization rights for content c in his control.
3. If there are, `P` creates Tx_6 and signs it.
4. `P` gives Tx_6 to `B`, `B` checks that it is correct, signs and gives Tx_6 back to `P`.
5. `P` sends a petition to `A` asking for the visualization key for c, by sending Tx_6 to the Bitcoin network.

The procedure can be repeated as many times as the buyer wishes, as far as there are any units of *AssetVisRights* left on his wallet.

5.3.5 | Artist serves decryption key

Once `A` receives the petition for the decryption key, that is, once he sees Tx_6 in the blockchain, `A`:

1. Validates the received transaction, Tx_6.
2. Sends the decryption key of the content, k, encrypted with the player's public key.

With respect to transaction validation, note that A receives Tx_6 through the Bitcoin network, and thus the transaction will already be a valid Bitcoin transaction. Therefore, A just checks the validity of the colored coins transaction and that one of the public keys in the output that Tx_6 is spending has authorization from a recognized manufacturer to operate (similarly than with Tx_4). A will then use the public key of the authorized player both to send the decryption key k and to encrypt it.

Then, A proceeds to encrypt the decryption key k with the public key of the player, $\hat{c} = E_{PK_{\mathtt{P}}}(k)$, and to send $\hat{c}$ to P, providing him with the key needed to decrypt content c. In order to offer transparency (that would allow to create fraud proofs when needed), A sends $\hat{c}$ also through the Bitcoin blockchain: he creates a transaction Tx_7 to the player's address encoding $\hat{c}$ as data (using, for instance, an OP_RETURN opcode), and sends the transaction to the Bitcoin network. Note that Tx_7 is a standard Bitcoin transaction, without colored coins data embedded into it.

This procedure is performed by A each time he receives units of *AssetVisRights* to one of his addresses.

5.3.6 | User visualizes content

Finally, once the player receives Tx_7, he proceeds to:

1. Decrypt $\hat{c}$.
2. Decrypt $\bar{c}$ and reproduce the content c.

In order to decrypt $\hat{c}$, the player uses the private key corresponding to the public key to which the key was sent, $k = D_{SK_{\mathtt{P}}}(\hat{c})$. With this procedure, P obtains the symmetric key k that allows to decrypt the content.

Once he learns k, the player can proceed to decrypt the content, $c = D_k(\bar{c})$. Finally, the player reproduces the content c and forgets k. As we will later explain in detail in Section 6, P has incentives to forget k because PM gets bitcoins for each visualization, so P does not want to reuse keys. Note that B (that indeed has incentive to reuse k) never knows k.

5.4 | Summary of the Bitcoin transactions involved in the system

Overall, eight different kinds of Bitcoin transactions (six of which carry also colored coins metadata) can be sent to the Bitcoin network during the operation of BArt. Table 11 summarizes the involved transactions, detailing whether they encode colored coins data, the asset they refer to, and the agents associated to the transaction.

Tx	Step	Type of tx	Involved asset	From	To
Tx_0	PM joins (Sec. 5.1.1)	Colored - issuance	*AssetPlayerAuth*	PM	PM
Tx_1	New P (Sec. 5.2.1)	Colored - transfer	*AssetPlayerAuth*	PM	P
Tx_2	PM revokes P (Sec. 5.2.2)	Colored - transfer	*AssetPlayerAuth*	P	PM
Tx_3	New c (Sec. 5.3.1)	Colored - issuance	*AssetVisRights*	A	A
Tx_4	B buys c (Sec. 5.3.2)	Bitcoin only	-	B&P	A
Tx_5	A sells c (Sec. 5.3.3)	Colored - transfer	*AssetVisRights*	A	B&P
Tx_6	B asks to vis. c (Sec. 5.3.4)	Colored - transfer	*AssetVisRights*	B&P	A
Tx_7	A serves c (Sec. 5.3.5)	Bitcoin (OP_RETURN)	-	A	P

TABLE 11 Summary of the Bitcoin transactions involved in the system.

6 | EVALUATION

In this section we first evaluate the security and privacy of our scheme, and then describe the overhead introduced by using BArt in each of the phases of the protocol.

6.1 | Security and privacy

First, we identify the threats BArt faces and the possible attacks any of the entities may try to perform. We describe the countermeasures BArt provides to each of the identified attacks.

Misbehaving **buyers** have a very limited set of actions since they do not know the decryption key k nor the decrypted content c at any stage of the protocol[#]. Indeed, they can try to record the content while it is being played (an attack common to any DRM scheme) or try to extract private keys from the player's internal memory. However, they can not double spend units of *AssetVisRights* if the artists waits for the adequate amount of confirmations nor try to obtain a decryption key without spending a unit of *AssetVisRights*.

Player manufacturers get revenue for each visualization performed by one of their players. Therefore, PM has incentives to follow the rules of the system. If any PM starts misbehaving, for instance, by not revoking *AssetPlayerAuth* to their misbehaving players, content producers may start removing the player manufacturer as a recognized PM for their contents (by not including it into the PM list in the metadata of their visualization assets).

Since **players** know the decryption key k whenever a content is reproduced in their system, misbehaving players would be able to keep the k for subsequent free reproductions of the content or even to sell k to third parties. However, on the one hand, players are produced by players manufacturers, that have high incentives to follow the rules of the system and thus are discouraged to program such behaviour into their players. On the other hand, an individual player may be hacked into performing actions not originally indicated by its player manufacturer. In this case, the player manufacturer will revoke its *AssetPlayerAuth*, making it unusable from that point on. This effectively forbids the players from that PM to reproduce any content associated to *AssetVisRights* issued after the misbehaviour has been detected. However, this does not affect *AssetVisRights* assets that have been already issued. In this case, two different situations may occur: 1) the *AssetVisRights* has already been sold to a buyer or 2) the *AssetVisRights* has been issued but not sold yet. In the first case, the PM has already been paid for his services, so it makes no sense to try to forbid reproduction. On the contrary, in the second case, the PM would still be able to obtain revenue when those assets are sold. One possible way to allow more control over this situation is to make *AssetVisRights* assets expire, so that the maximum amount of time a PM may still obtain revenue once it has misbehaved is delimited by the A himself. This can be done by adding an expiration policy to transaction Tx_3 (Table 8). Apart from keeping the encryption key k for further free reproductions, Players may also misbehave by not reproducing the decrypted content to the buyer once they receive the decryption key k. Again, since PM gets revenue for each visualization, players are incentivized to show content to the buyer, so that the buyer keeps using them as his player.

Artists may try to subvert the system by not sending *AssetVisRights* in response to a buyer's payment. Because the Bitcoin blockchain offers transparency (all transactions are public), such behaviour would be detected and publicly noticeable by any observer. If a A does not respond to a buyer payment, a proof of that behaviour will remain in the blockchain forever (a transaction paying for and requesting the asset will remain in the blockchain whereas no assets will be transferred back to the payee). Similarly, artists may also try to subvert the system by sending a corrupted decryption key. Again, any player that detects this behaviour will be able to construct a fraud proof, that is, a proof that anyone can verify that the artist tried to cheat. Let $h_1 = H(k)$, $h_2 = H(\overline{c})$ be, respectively, the hash of the decryption key and the hash of the encrypted content the artist committed to on asset issuance (recall that these values can be found in the metadata of the respective *AssetVisRights*). Moreover, $\overline{c}$ is the publicly accessible encrypted content. The first way A may try to cheat is by sending to B a corrupted key $\hat{c}' = E_{PK_{\mathrm{P}}}(k')$ (with $k \neq k'$). Then, since $\hat{c}'$ is published in the blockchain, the player may reveal his private key, demonstrating that $D_{SK_{\mathrm{P}}}(E_{PK_{\mathrm{P}}}(k')) = k' \neq k$. Recall that all the information is publicly available in the blockchain, and thus the fraud proof is accessible by any observer. The second way A may try to cheat is by committing to a corrupted key k during asset issuance. Again, it is possible to demonstrate that the artist cheated by using information in the blockchain. Indeed, the player may present the decryption key k he has obtained. Showing that $H(\mathrm{k}) = h_1$ (and thus $\mathrm{k} = k$), $H(\overline{c}) = h_2$, $D_k(\overline{c}) = c'$ and that c' is invalid, the player has a fraud proof.[‖]

Considering that all colored coins transactions in our scheme are public, **external observers** may also try to subvert the system with the information they obtain by analyzing the blockchain. However, as far as the entities' private keys remain secret,

[#]Note that, in our scheme, the need for Players arise from the fact that Artists can not directly provide the decryption key to Buyers, because the first time a Buyer acquires rights to visualize a content he could be able to store the decryption key, and further use it without never paying the Artist again. By introducing Players, the decryption key is never revealed to the Buyers, so this behaviour is prevented. In turn, Players have incentives to forget the decryption key, because their manufacturers get revenue from each reproduction, and because they risk losing authorization to reproduce content if they misbehave (and thus they lose revenue).

[‖]BArt offers protection from misbehaving Artists by allowing to generate proofs of fraud, that can be used to report any misbehaviour. If we want to go one step further and prevent malicious behaviour from even happening, we could try to integrate a fair data selling protocol within BArt. The fair exchange protocol from Delgado et al.[23] allows buyers to ensure the content is genuine and the exchange (content for payment) is atomic over the Bitcoin blockchain, and thus will be a perfect fit for BArt.

	Computing time	Network data
New PM (Sect. 5.1.1)	Tx0 gen.	send Tx0
New A (Sect. 5.1.2)	-	-
New B (Sect. 5.1.3)	-	-
Add P (Sect. 5.2.1)	Tx1 gen.	send Tx1
Revoke P (Sect. 5.2.2)	Tx2 gen.	send Tx2
New c (Sect. 5.3.1)	content encryption hash content Tx3 gen.	upload content send Tx3
B pays (Sect. 5.3.2)	Tx4 gen.	send Tx4
A sells (Sect. 5.3.3)	Tx5 gen.	-
B asks to vis. (Sect. 5.3.4)	Tx6 gen.	-
A serves key (Sect. 5.3.5)	encrypt k Tx7 gen.	send Tx7
B visualizes (Sect. 5.3.6)	decrypt k descrypt content	download content

TABLE 12 Resource-consuming operations in each of `BArt`'s actions.

external observers will not be able to steal assets nor to decrypt content keys thanks to the usage of cryptography. Nonetheless, one of the existing threats is to take advantage of the fact that all transactions are public to try to compromise buyers' privacy. During the player setup phase, `PM` creates a transaction Tx_1 to transfer player authorization rights to addresses of the same player `P`. In order to avoid linkability between the different addresses of the same player, which would allow an external observer to construct a profile of the visualization preferences of a certain (unknown) user, the `PM` can mix different players addresses in the same transaction and/or distribute addresses from the same player into different transactions. By doing so, one achieves protection in front of external observers. Note, however, that since the player manufacturer is in possession of the master seed from which the player's wallet was derived, the `PM` is still able to link different addresses of the same player.

6.2 | Overhead

Because the proposed scheme relies on the Bitcoin blockchain, it can be implemented using very few resources. In this section, we evaluate the computing resources needed to execute the different phases of `BArt`. Specifically, we have evaluated each resource-consuming action in terms of execution time and amount of data to send through the network. Table 12 summarizes the most resource intensive operations for each of the actions of our protocol.

6.2.1 | Network data

The size of the Bitcoin transactions involved in `BArt` is rather small, so sending these transactions over the network produces a small overhead. In this section, we provide the sizes of the Bitcoin transactions in `BArt` to show they introduce little overhead over the system.

Regardless of the `OP_RETURN` outputs, we can classify the Bitcoin transactions in `BArt` in three cases:

1. One P2PKH input and one P2PKH output (Tx_0, Tx_1[**], Tx_2, Tx_3, Tx_7).
2. One P2PKH input and one 2-out-of-2 multisig output (Tx_5).
3. One 2-out-of-2 multisig input and one P2PKH output (Tx_4, Tx_6).

Then, some of these transactions will have an additional `OP_RETURN` output carrying colored-coins metadata. Let us assume that all colored coins transactions in `BArt` use the maximum standard available data space in its `OP_RETURN` outputs (80 bytes).

[**] This would correspond to the worst case scenario where a transaction is used to authorize only one player. In the general case where multiple players are authorized by a single transaction, the transaction will be bigger, but the size per authorization asset will be much smaller.

Type	1	2	3
Version		4	
Number of inputs		1	
Outpoint		36	
Script len.		1	
Script	107	107	145
nSequence		4	
Number of outputs		1	
Amount		8	
Script len.		1	
Script	25	71	25
nLockTime		4	
Total without op_return	192	238	230
Total with op_return	274	320	312

TABLE 13 Sizes (in bytes) of Bitcoin transactions in our scheme.

Note that only colored-coins issuance transactions would need that much space, whereas transference transactions wouldn't use it. The script of an `OP_RETURN` output with 80-byte data is 82-bytes long.

The final size of each transaction will thus be determined by the scripts in its inputs and outputs, and the additional existence of an `OP_RETURN` output. Table 13 summarizes the sizes of transactions in our scheme. For instance, transaction Tx_1 (one P2PKH input and one P2PKH output, with colored coins data) will be 274 bytes, whereas Tx_4 (one 2-out-of-2 multisig input and one P2PKH output, without colored coins metadata) will be 230 bytes.

Therefore, in any case, the amount of data that has to be sent over the network for Bitcoin transactions is very small.

Beyond the size of transactions that have to be sent to the Bitcoin P2P network, the four entities described in our scheme need to have a bitcoin wallet capable of handling colored coins transactions: the artist and the player manufacturer will need to issue and transfer assets, and the player and the buyer will need to transfer assets. Note that having a wallet that understands the colored coins protocol is the only requirement for any of the entities to participate into the system. Therefore, the amount of traffic needed to maintain a wallet must also be considered when computing the overhead of our system. This amount of traffic is highly variable, and depends on the chosen kind of wallet as well as the specific configuration it uses. For instance, a SPV client will need to download block headers, which means 80-bytes each 10 minutes. Therefore, a SPV client downloads just a few kilobytes of data per day. On the other hand, a bitcoin core full-client configured with `blocksonly` requires around 150MiB of incoming data and 1 MiB of outgoing data per day. Finally, a full-node accepting incoming connections and downloading and relaying transactions may download around 1GB of data, and need to upload 20GB per day.

With respect to connectivity, the only entity that should be continuously online (i.e., connected to the Bitcoin network) is the artist: he should be able to sell visualization assets and serve decryption keys whenever a buyer or player requests so. On the contrary, player manufacturers do not need to be online at all. They just need to connect to the Bitcoin network when they issue authorization assets and when they transfer them (on new player setup). Note that there is no need to contact `PM` in order to validate the ownership of authorization assets: all the information needed is stored in the Bitcoin blockchain. There is no need also for the `PM` to be online to be able to receive payments. With respect to players and buyers, they only need to connect to the Bitcoin network whenever they want to buy new visualization assets or exchange already owned assets for the decryption keys.

Finally, note that the Artist also needs to upload the encrypted content before being able to distribute it. The amount of data to upload will depend on the size of the encrypted content. In any case, this does not introduce any overhead over any other content distribution system, since the content must always be uploaded at least once.

6.2.2 | Execution time

In order to evaluate the computational needs of the actions in our protocol, we have analyzed the time needed to run them in an Intel(R) Core(TM) i7 CPU @ 1.80GHz laptop with 8 cores, 8GB DDR4 RAM, and a SSD disk. We have used our open-source toolkit `bitcoin-tools`[††] to generate Bitcoin transactions and to sign them. Note that this library is not optimized for performance, so results showed here could surely be improved. We have executed each action 10 times, and report here the mean and standard deviation of the 10 executions.

`BArt` can be used with artworks of diverse nature, and thus the size of the content c an Artist incorporates into the scheme may range from a few megabytes of data (e.g. a song) up to gigabytes of data (e.g. a full-HD movie). Whenever the content is large, the operations done over this content will be the ones taking the most computing time. This involves both encrypting/decrypting the content and hashing it. Figure 3 shows the user and system time needed to encrypt, decrypt or hash content files of different sizes. Hashing is the operation requiring the most time, but it is only performed once for each content (when the Artist issues *AssetVisRights*), and it takes less than 1.5 minutes even for large files (8GB of data).

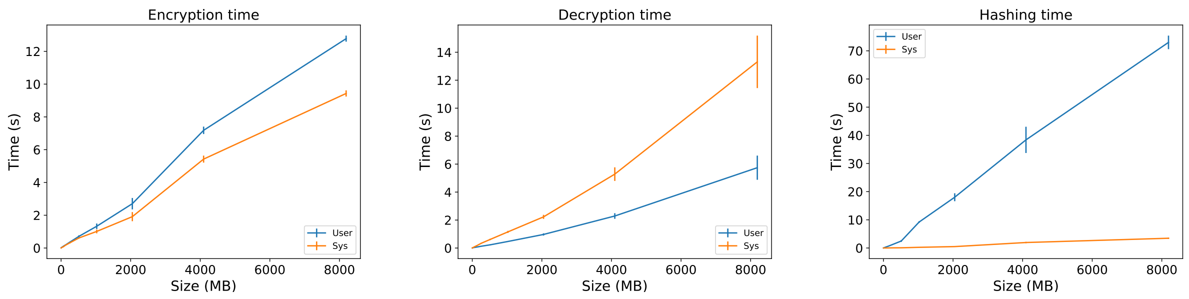

FIGURE 3 Computing time spent to encrypt, decrypt, or hash content files of different sizes.

The other kind of computing-intensive actions in `BArt` is transaction generation. Generating the Bitcoin transactions from `BArt` involves computing the hash of the transaction data and signing that hash. The computing time required to do so is almost negligible. The mean time to sign a one P2PKH-input one P2PKH-output transaction is 0.026 seconds (with a std of < 0.001), whereas it slightly increases to 0.028 seconds (and 0.004 of std) for a one 2-out-of-2 Multisig-input and one 2-out-of-2 Multisig output transaction. Note that there is no such transaction in our scheme (transactions have either a multisig input or a multisig output, but never both), but we have used this as an upper limit.

Beyond the computational costs directly associated to `BArt` there are also computational requirements regarding the cost of running the bitcoin client. Again, these costs will highly depend on the type of client being run. SPV clients costs are low, whereas running a full bitcoin client has stronger requirements. Even for full clients, the hardware of a Raspberry Pi is enough to run them.

7 | CONCLUSION

A widely known problem the entertainment industry faces is the proper attribution and enforcement of rights to artists, together with the monetization of their artworks. Previous studies performed by Marcus O'Dair et al.[6] have claimed that blockchain technologies have transformative potential for certain digital artworks, for instance, for recorded music. In their study, Marcus O'Dair et al. identify four areas in which blockchains may be able to solve problems the current music ecosystem has:

1. Provide a networked database for music copyright information.
2. Facilitate fast, frictionless royalty payments.
3. Offer transparency through the value chain.

[††]https://github.com/sr-gi/bitcoin_tools/

4. Provide access to alternative sources of capital.

`BArt` indeed contributes in all these four areas and, additionally, presents a solution to one of the areas not mentioned in the study: the enforcement of the rights the artists grant over their artworks. `BArt` provides a mechanism for managing digital artworks, serving as a networked database for copyright information, where the database corresponds to the Bitcoin blockchain with colored coins data embedded. Moreover, the system facilitates royalty payments, made through transactions in the Bitcoin system. The use of the Bitcoin blockchain offers transparency through the publication of both artworks' metadata and protocol information. Such transparency ensures the proper behavior of all parties interacting in the system, since actions are recorded and can be later retrieved to prove its correctness. Finally, `BArt` also enforces the usage rights authors set for their artworks using an open vision where, not only special devices but any player is encouraged to follow the usage rights model through an economic incentive scheme. Such open model, where the artists do not need neither an expensive infrastructure nor a direct control of special devices with DRM capabilities, provides new artists, that have just started their career, with an alternative source of capital (since they can directly monetize their artwork).

Further research faces two different approaches. On one hand, in a theoretical plane, the system could be enhanced with its integration with the lightning network in order to provide more flexibility to the payments. On the other hand, from a practical point of view, a proof of concept of the proposed system could be developed using the Bitcoin Testnet network.

ACKNOWLEDGEMENTS

This work is partially supported by the Government of Catalonia (2017 SGR 00705). Cristina Pérez-Solà was funded by the Martí i Franquès research grants programme (2017PMF-POST2-06) during part of the time when this work was done. The authors also gratefully acknowledge support from the Spanish Government, under grants RTI2018-095094-B-C22 CONSENT and TIN2014-57364-C2-2-R SMARTGLACIS.

BIBLIOGRAPHY

References

1. McConaghy T, Pon B, McConaghy M. Ascribe: Lock in attribution, securely share and trace where your digital work spreads. https://www.ascribe.io/; Last accessed: September 2018.
2. McConaghy T, Holtzman D. Towards An Ownership Layer for the Internet. tech. rep., ascribe GmbH; 2015.
3. Jonghe dD, McConaghy T. SPOOL Protocol. https://github.com/ascribe/spool; May 2015, Last accessed: September 2018.
4. Mediachain Labs . Mediachain. https://blog.mediachain.io/introducing-mediachain-a696f8fd2035; January 2016, Last accessed: September 2018.
5. Mediachain Labs . Mediachain Attribution Engine: Find great images and reward creators. http://images.mediachain.io/; Last accessed: September 2018.
6. O'Dair M, Beaven Z, Neilson D, Osborne R, Pacifico P. Music on the blockchain. tech. rep., Middlesex University; 2016.
7. Kitahara M, Kawamoto J, Sakurai K. A method of digital rights management based on Bitcoin protocol. In: ACM. ; 2014: 84.
8. Fujimura S, Watanabe H, Nakadaira A, Yamada T, Akutsu A, Kishigami JJ. BRIGHT: A concept for a decentralized rights management system based on blockchain. In: IEEE. ; 2015: 345–346.
9. Kishigami J, Fujimura S, Watanabe H, Nakadaira A, Akutsu A. The Blockchain-Based Digital Content Distribution System. In: IEEE. ; 2015: 187–190.
10. JPEG Group . JPEG explores blockchain in its standards. .

11. Temmermans F, Bhowmik D. JPEG White paper: Towards a Standardized Framework for Media Blockchain. tech. rep., ISO/IEC JTC 1/SC29/WG1 (ITU-T SG16); 2018.

12. Narayanan A, Bonneau J, Felten E, Miller A, Goldfeder S. *Bitcoin and cryptocurrency technologies*. Princeton University Press . 2016.

13. Antonopoulos AM. *Mastering Bitcoin: Unlocking digital cryptocurrencies*. O'Reilly Media, Inc. . 2014.

14. Higgins S. Overstock to Issue Digital Corporate Bond on Bitcoin Blockchain. http://www.coindesk.com/overstock-to-issue-digital-corporate-bond-on-bitcoin-blockchain/; June 2015, Last accessed: September 2018.

15. Palaniappan R, Taylor R. Origin: The Primary Marketplace. https://originmarkets.com/; March 2015, Last accessed: September 2018.

16. Nazare J, Hamilton K, Schmidt P. Digital Certificates Project. http://certificates.media.mit.edu/; Last accessed: September 2018.

17. Ivgi N. BTProof: Trusted timestamping on the Bitcoin blockchain. https://www.btproof.com/; 2013, Last accessed: December 2016.

18. Shomer A. The Colored Coins Protocol. https://github.com/Colored-Coins/Colored-Coins-Protocol-Specification; Last accessed: September 2018.

19. Gundry D, Charlon F. Open Assets Protocol. https://github.com/OpenAssets/open-assets-protocol; 2016, Last accessed: September 2018.

20. ChromaWay . EPOBC: Enhanced padded order-based coloringe. https://github.com/chromaway/ngcccbase/wiki/EPOBC_simple; 2014, Last accessed: September 2018.

21. Greenspan G, Rozantsev M. CoinSpark - Coin Sciences Ltd. http://coinspark.org/developers/; Last accessed: February 2017.

22. Wuille P. BIP 0032: Hierarchical Deterministic Wallets. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki; February 2012, Last accessed: September 2018.

23. Delgado-Segura S, Pérez-Solà C, Navarro-Arribas G, Herrera-Joancomartí J. A Fair Protocol for Data Trading Based on Bitcoin Transactions. *Future Generation Computer Systems* 2017. doi: https://doi.org/10.1016/j.future.2017.08.021

APPENDIX

A - SYSTEM FLOW CHARTS

This appendix contains the flow charts of the actions that can be performed by all the actors of the system, that is, the buyers (Figure A1), the player manufacturers (Figure A2), and the artists (Figure A3).

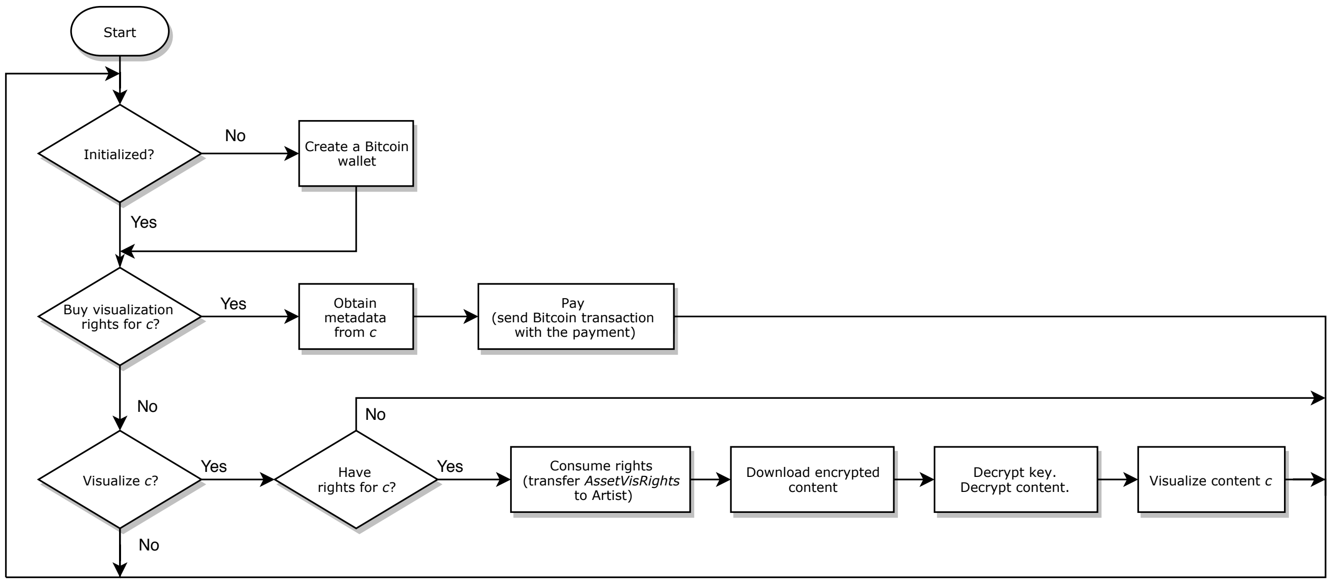
Start
Initialized?
No
Create a Bitcoin wallet
Yes
Buy visualization rights for c?
Yes
Obtain metadata from c
Pay (send Bitcoin transaction with the payment)
No
Visualize c?
Yes
No
Have rights for c?
Yes
No
Consume rights (transfer AssetVisRights to Artist)
Download encrypted content
Decrypt key. Decrypt content.
Visualize content c
No

FIGURE A1 Flow chart of the actions that can be performed by the buyers.

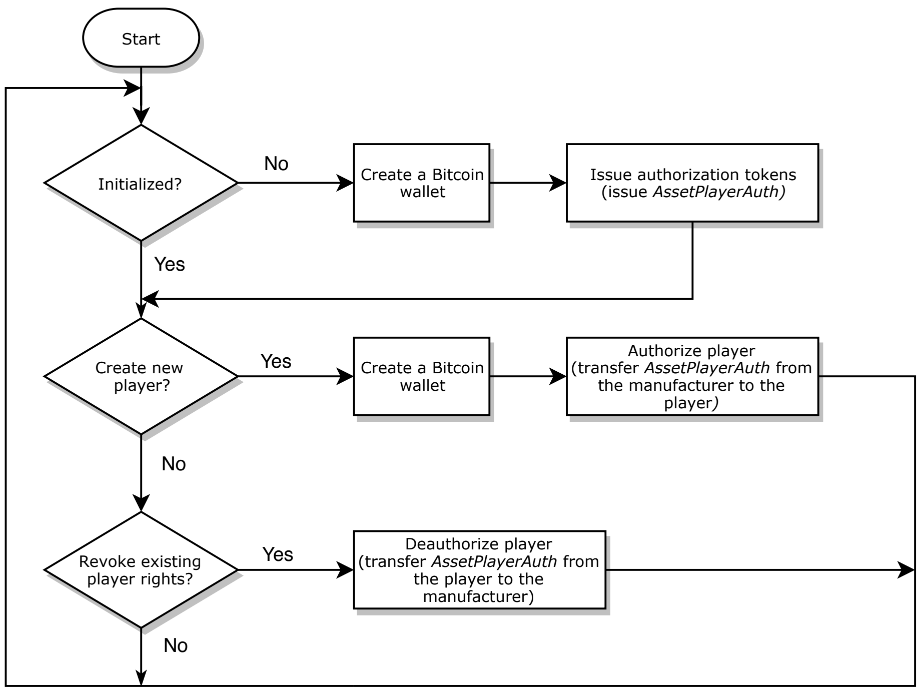
Start
Initialized?
No
Create a Bitcoin wallet
Issue authorization tokens (issue AssetPlayerAuth)
Yes
Create new player?
Yes
Create a Bitcoin wallet
Authorize player (transfer AssetPlayerAuth from the manufacturer to the player)
No
Revoke existing player rights?
Yes
Deauthorize player (transfer AssetPlayerAuth from the player to the manufacturer)
No

FIGURE A2 Flow chart of the actions that can be performed by the players' manufacturer.

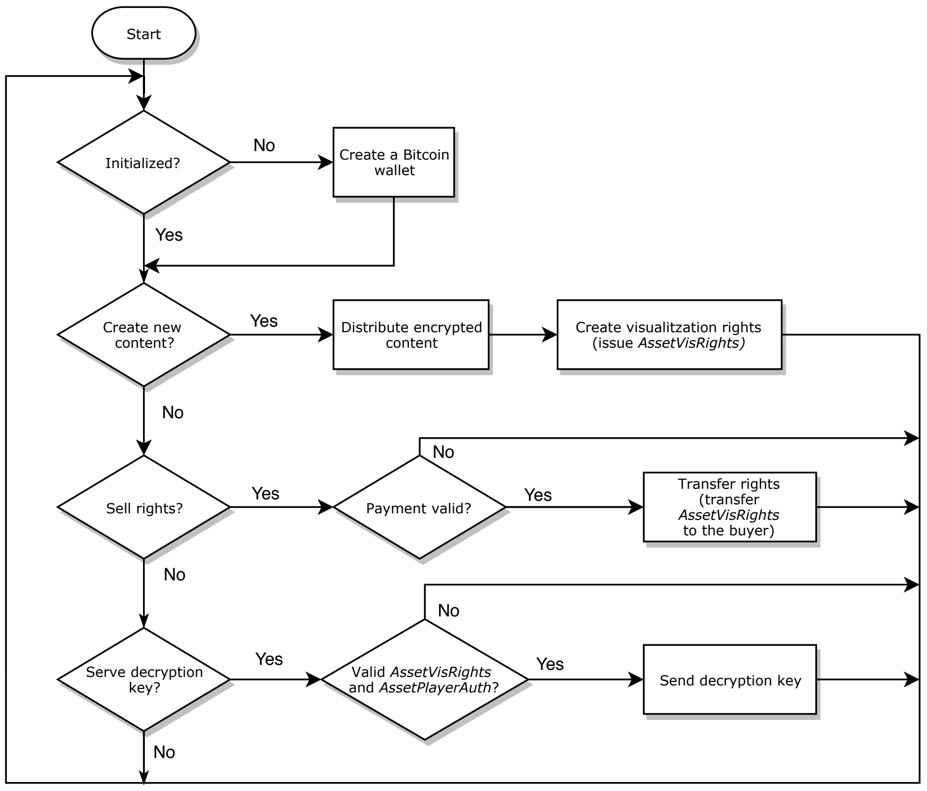
Start
Initialized?
No
Create a Bitcoin wallet
Yes
Create new content?
Yes
Distribute encrypted content
Create visualitzation rights (issue AssetVisRights)
No
Sell rights?
Yes
Payment valid?
No
Yes
Transfer rights (transfer AssetVisRights to the buyer)
No
Serve decryption key?
Yes
Valid AssetVisRights and AssetPlayerAuth?
No
Yes
Send decryption key
No

FIGURE A3 Flow chart of the actions that can be performed by the artists.