WATERMARK NFT

A NFT ownership certification process and norm using steganography for digital watermarking.

Clément Stauner & Benjamin Becerra

Co-editor Christophe OZCAN

Table of Contents

Versions

Version	Title	Description	date
1.0	Watermarking NFT files	Describes the technology and its advantages compared to the current solutions.	2021-12-05
2.0	Ownership-Chain	Introduction of the Ownership-Chain and revises terms' names.	2022-02-28
2.1	Graphical improvement	Added graphics in-between paragraphs for better understanding.	2022-05-31

Introduction

A non-fungible token (NFT) is a unique token representing an anchored unit of data stored in a digital distributed ledger (blockchain). NFT are a part of an associated reproducible file such as photos, videos, audios, and other types of digital files considerate as unique items (like certificates of authenticity) and use blockchain technology to provide a public proof of ownership.

A copy of the original file is not limited to the NFT owner and can be copied and shared like any other file. The lack of interchangeability (fungibility) distinguishes NFTs from blockchain cryptocurrencies such as Bitcoin and Ethereum which are fungibles.

The first NFT project, Cryptokitties took place on the Ethereum blockchain in 2015 and has since grown into an important sector of the Western art industry.

Centralized File Storage

The current architecture of NFTs usually do not store files on the blockchain due to the Blockchain capabilities to not store important data size . The token acts like a certificate of ownership and is linked to a URL address that points to the tokenized file in question, but the original file remains subject to « link rot ».

A link rot (also called link death, link breaking, or reference rot) are hyperlinking phenomena that cause this file source to move to a new URL address and/or becoming potentially a permanent unreachable link. No centralized service could be available forever. URL links that no longer point to a destination are often referred to as broken or dead links (or orphaned links)

In the past, some NFTs were permanently disabled due to the use of non-redundant off-chain storage. Blockchain network (like Ethereum) have limited storage capacity (few kilo bytes). For that reason, an on-chain NFT network where all metadata medias are referenced directly in the associated token smart contract which is not common because the data is currently stored on off-chain network (AWS server, IPFS, etc.).

Developing an on-chain NFTs network could be completely secure, more difficult to hack and be always reachable online. Compared to an off-chain solution with higher censorship risk (e.g., off-chain metadata can be altered, lost to a project that goes bust, censored by AWS, etc.).

To counter this phenomenon, files are usually stored in a decentralized and secure file system such as IPFS (Interplanetary File System). The Interplanetary File System is a peer-to-peer network for storing and sharing data in a distributed way. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices. This hybrid approach provides developers with ease of use and flexibility.

Because tokens are functionally separate from the underlying file, anybody can easily save a copy of an NFT's file, popularly through a right click. NFT supporters disparage this duplication of NFT artwork as a "right-clicker mentality", with one collector comparing the value of a purchased NFT to that of a status symbol "to show off that they can afford to pay that much".

The "right-clicker mentality" phrase spread virally after its introduction, particularly among those that were critical of the NFT marketplace who used the term to flaunt the ability to capture digital art backed by NFT with ease.

Problem Definition

Plagiarism and fraud

We have read many examples of artists having their work copied without permission and sold as an NFT. After the artist Qing Han died in 2020, her identity was assumed by a fraudster and several her works became available for purchase as NFTs. Even if the piece is already available as an NFT, a fraudster can “mint” a new NFT linking to the same file using the same hypertext link.

Similarly, a seller posing as Banksy succeeded in selling an NFT supposedly made by the artist for $336,000 in 2021, with the seller in this case refunding the money after the case drew media attention.

A process known as "sleep minting" can also allow a fraudster to mint an NFT in an artist's wallet and transfer it back to their own account without the artist becoming aware. This allowed a white hat hacker to mint a fraudulent NFT that had seemingly originated from the wallet of the artist Beeple.

Problem root causes

One of the main problems of NFTs is that the original file link between the token smart contract certificate and the hosted file storage system is unidirectional. The only thing linking them is a URL link stored as a string in the smart contract metadata. If the link dies, the data source is changed or a fraudulent actor could execute a counterfeit smart contract pointing to same URL Link, the integrity of the NFTs is undermined.

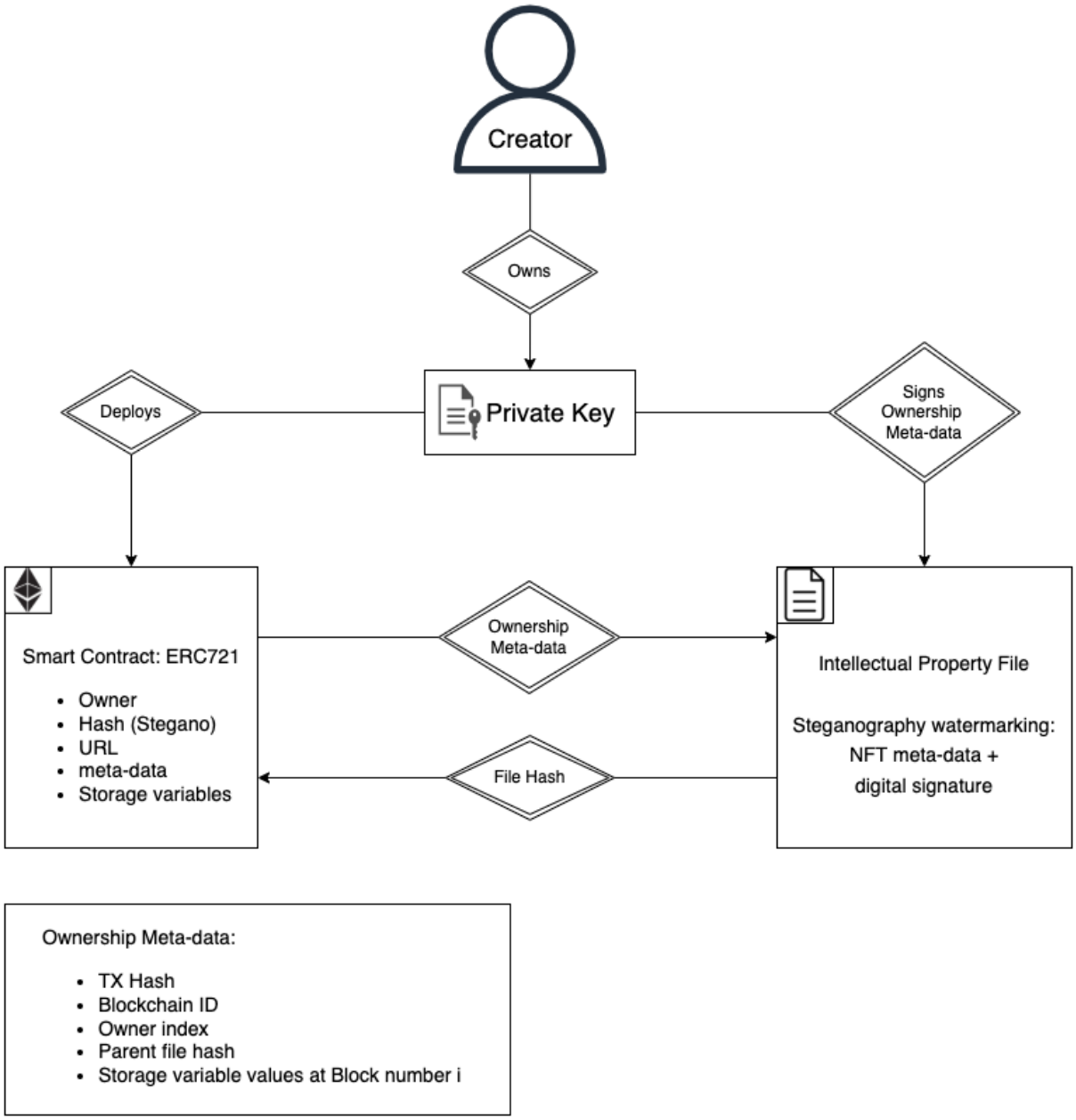
To solve this problem, the approach is to build a solution that guarantees the integrity of the entire NFT metadata. Each person has their own unique style of handwriting, whether it is everyday handwriting or their personal signature. Cultural environment and the characteristics of the written form of the first language that one learns to write are the primary influences on the development of one's own unique handwriting style. Even identical twins who share appearance and genetics do not have the same handwriting. Because each person's handwriting is unique and different, it can be used to verify a document’s writer.

Moreover, for NFTs to be adopted by institutions, the solution needs to guarantee the integrity of the NFT’s metadata as previously described.

The solution needs as well to be open, transparent, and easily auditable to be adopted by platforms and used just like a person opening a JPEG file on his computer.

High Level Solution

Schema



Explanation

Our solution incorporates two concepts. The first one, the file needs to have the handwriting/brushstroke embedded in the data, just like the real Mona Lisa has the Leonardo da Vinci brushstroke inside its painting. Second, the link between the smart contract and the file is reinforced using existing cryptography and is made bidirectional.

The idea is to have the proof of creation of some file by a creator without having to verify with the concerned individuals (real creator or fraudster). There needs to be a signed statement “I, creator name XXX certify that I created this content ABCDEF.” embedded in the file. Only the true creator could have signed this statement and no one else. We already have a technology enabling us to do just that in the Blockchain ecosystem. It is the public key encryption technology that permits to sign messages to certify the authenticity.

To embed the data, we could put it in the file meta-data, but it would not be secured. There is a technique called steganography to hide data in the file’s content. Steganography is the practice of concealing a message within another message or a physical object. In computing/electronic contexts, a computer file, message, image, or video is concealed within another file, message, image, or video. Media files are ideal for steganographic transmission because of their large size. For example, a sender might start with an innocuous image file and adjust the color of every hundredth pixel to correspond to a letter in the alphabet. The change is so subtle that someone who is not specifically looking for it is unlikely to notice the change.

The larger the cover message (in binary data, the number of bits) relative to the hidden message, the easier it is to hide the hidden message (as an analogy, the larger the "haystack", the easier it is to hide a "needle"). So digital pictures, which contain much data, are sometimes used to hide messages on the Internet and on other digital communication media. In our case, it is not to “hide” to message sickly speaking but to embed a watermark certifying the file’s ownership.

Steganography and watermarks look similar, but they are not. In steganography, hidden messages should be left untouched until they reach their destination. Steganography can also be used for watermarks whose message (just an identifier) is hidden in the image, so that its source can be tracked or verified, and the image can also be identified. In such cases, the technique of hiding the message (here, the watermark) must be robust to prevent tampering. However, watermarks may require vulnerable watermarks that can be easily modified to ensure that the image has been tampered with. This is the main difference between steganography and watermarking.

In our case, this could translate to the following process: The artiste creates his file and a smart contract to certify the ownership. The meta-data signed by the owner are called the ownership meta-data (different for every new owner of the token). It contains: the owner’s public key, the owner counter/index, the blockchain network ID, the hash of the blockchain transaction and the “parent file hash” (here the genesis hash).

The signature must be done after the smart contract deployment on the Blockchain. So that the transaction hash of the contract creation can be included in the meta-data. This result with a link from the file to the smart contact (first half of the bi-directional link between the NFT smart contract and the file).

Optionally, other storage variable values at block number index X (at the time of the ownership certification) can be included if judged necessary. The value of these storage variables must be at the block number of the transaction hash specified above. The other values included in the meta-data must be on the smart contract or elsewhere on the blockchain (e.g., other smart contract, oracles, etc.) to certify their authenticities. Because if they are not, there is no way to trace the provenance of their values during an audit.

For instance, if the token represents social media post (YouTube video, tweet, Instagram image, etc.) containing public meta-data evolving overtime (like, share, comments count, view count, etc.) and those meta-data are included in the smart contract or elsewhere on the blockchain), if the developer judges relevant to include the storage variable values at the time of the ownership certification, it can be done.

The token owner signs ownership meta-data with his private key from the Blockchain ecosystem, he embeds the signed meta-data in the file using steganography. The resulting steganographied file (original file + signed statement data) will be the file used as an NFT, not the original file. The original file is used to generate every new stegano file, for every new owner of the token.

The resulting NFT is one that has a signed message embedded in the content using the private key of the file's creator and a unique hash in the smart contract. This NFT can be now stored in a safe place like IPFS or another place if needed.

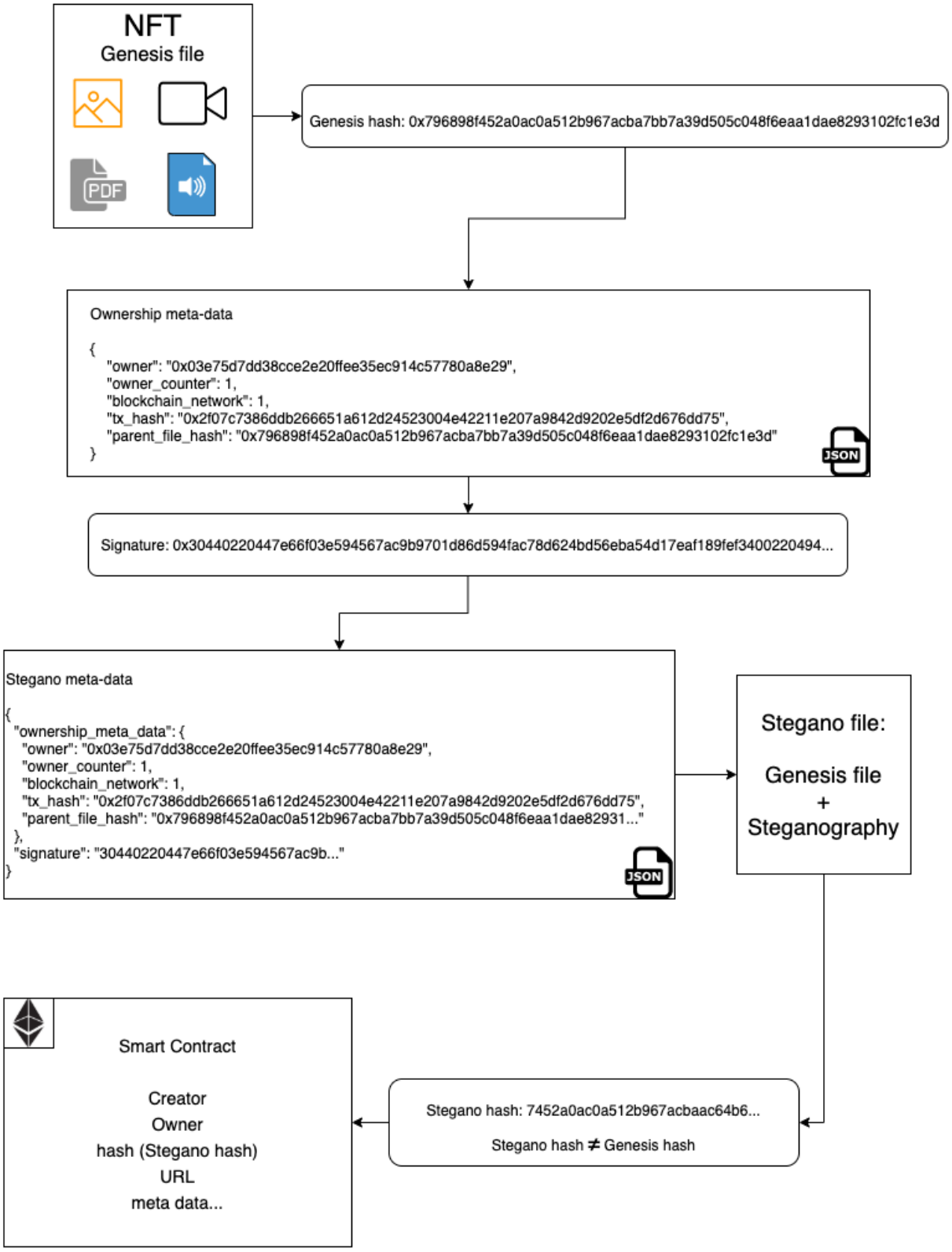
The last step of the process will be to compute a second hash. This time the hash of the steganographied file and store it in the NFT smart contract. Bear in mind that the second hash is different from the parent hash because the input file is different. The hash needs to be stored in the smart contract as a storage variable to secure the link between the file and the smart contract. This result with a second link but this time from the file to the smart contract (second half of the bi-directional link between the NFT smart contract and the file).

Concerning the bi-directional link between the NFT smart contract and the file. First, there is the ownership certification transaction hash embedded in the file using steganography, thus resulting with a unidirectional link from the smart contract to the file. Second, there is the stegano hash stored on the smart contract, thus resulting in another unidirectional link but this time from the file to the smart contract. The outcome of the process is a pseudo bi-directional link between the NFT and the file that strengthen the relationship between the two.

Solutions Details

Token Minting

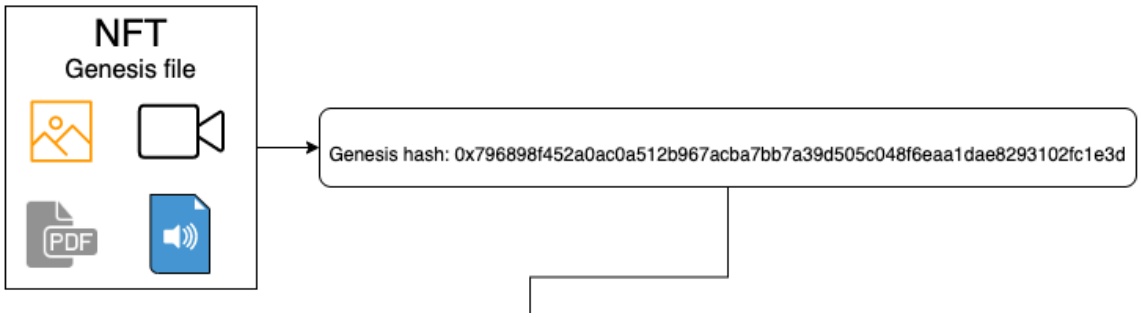
Schema



Explanation

File and meta-data

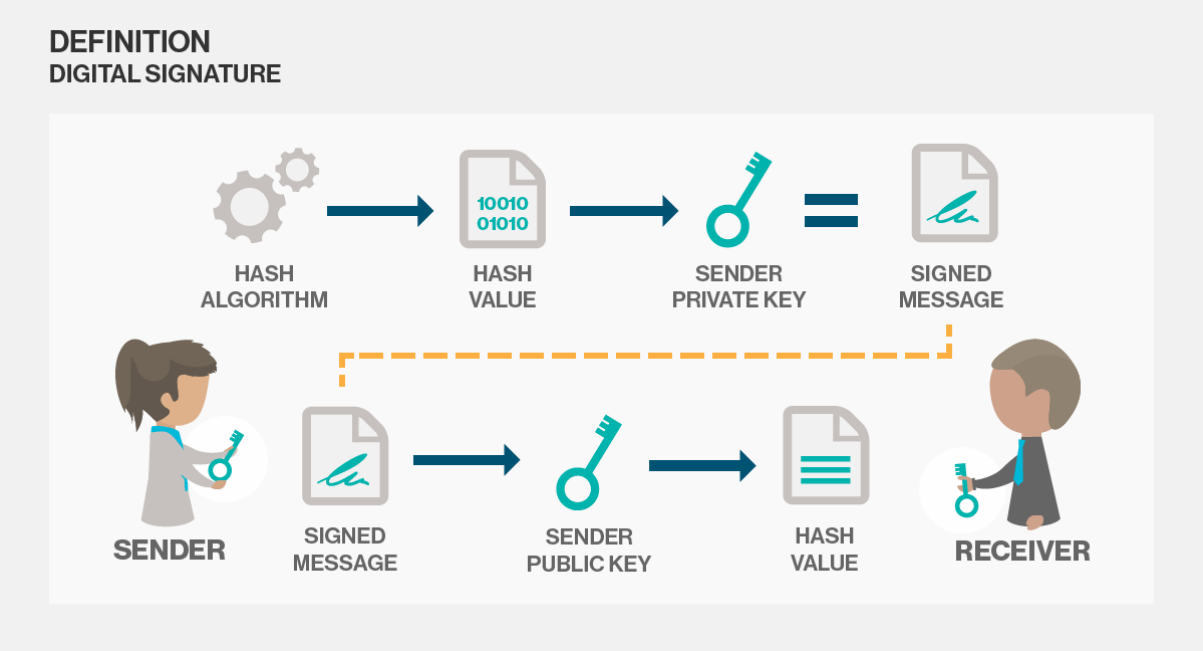
The file (image, audio, video, PDF) is generated by the creator. The file's hash must be computed. The hash of the genesis file is compulsory representing the original digest of the file called the genesis hash. For the hash function, we recommend using at least the SHA2 family algorithm, but you can customize with any other hash algorithms (SHA-256, KECCAK-256, etc.).



The ownership meta-data contains the owner's public key, the owner counter/index, the blockchain network ID, the hash of the blockchain transaction, the parent file hash and optionally storage variable values (not included in the schema). The values of the storage variables put in the ownership meta-data must be found in the smart contract storage variables at the block of the transaction hash specified.

The resulting ownership meta-data is in JSON format. The genesis meta-data are signed by the file creator using his signature scheme through his own wallet (the public key which he will use to mint the NFT). The NFT creator signs the message with his private key to create a short digital signature on the message.

Anyone with the proper public key of the NFT creator can verify this message with the alleged digital signature. If the signature matches the message, the origin of the message is verified (it must have been created by the owner of the corresponding private key).



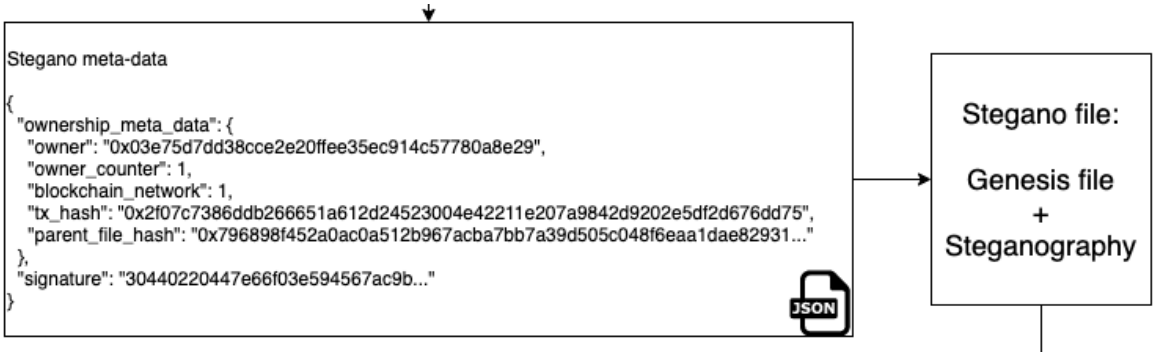
Like the actual blockchain ecosystem, a user certify that he has done a transaction using his private key: “I, Alice, certify I will send 1 bitcoin to Bob, at this timestamp”. No one can emulate this signature, and everyone can verify it by using the signatory public key. So, this will translate to: “I, Alice, certify that I have created a file with this hash and other meta-data”. This system creates non-repudiation to ensure that one party cannot successfully dispute its authorship of a document or a message.

The resulting signature will be included in a new JSON with the new meta-data. This JSON is called the steganography meta-data (or simply stegano meta-data) because it is the message that will be embedded in the genesis file using the steganography algorithm creating the steganographied file (stegano file). This JSON is composed by at least two fields: the genesis meta-data and its signature.

Stegano meta-data

```
{
  "ownership_meta_data": {
    "owner": "0x03e75d7dd38cce2e20ffee35ec914c57780a8e29",
    "owner_counter": 1,
    "blockchain_network": 1,
    "tx_hash": "0x2f07c7386ddb266651a612d24523004e42211e207a9842d9202e5df2d676dd75",
    "parent_file_hash": "0x796898f452a0ac0a512b967acba7bb7a39d505c048f6eaa1dae82931..."
  },
  "signature": "30440220447e66f03e594567ac9b..."
}
```

This steganography meta-data is embedded in the genesis file using a steganography algorithm (could be any algorithm but needs to be specified in the smart contract for future decode/audit). There is no need to use a passphrase to encode the stegano meta-data inside the file because we do not want to hide it in the literal sense. Everyone knows there is a JSON embedded inside the file data. There is no need to over engineer the process.



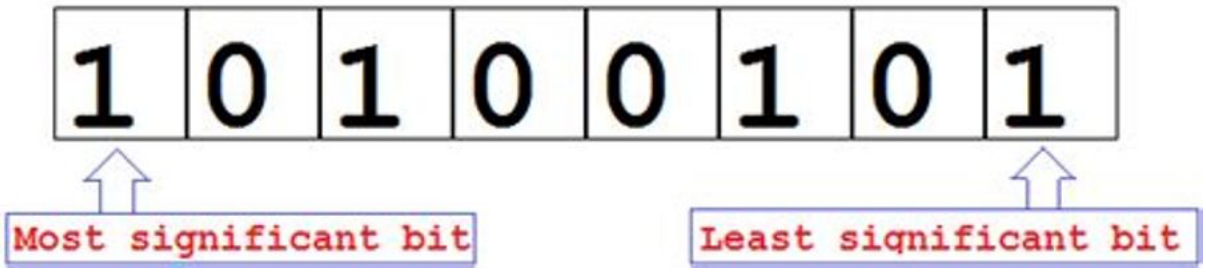
If some external actor wants to audit the NFT, he will use the same steganography algorithm as the creator. He will retrieve the steganography meta-data (JSON) from the stegano file.

The genesis file cannot be retrieved by applying the steganography algorithm on the stegano file.

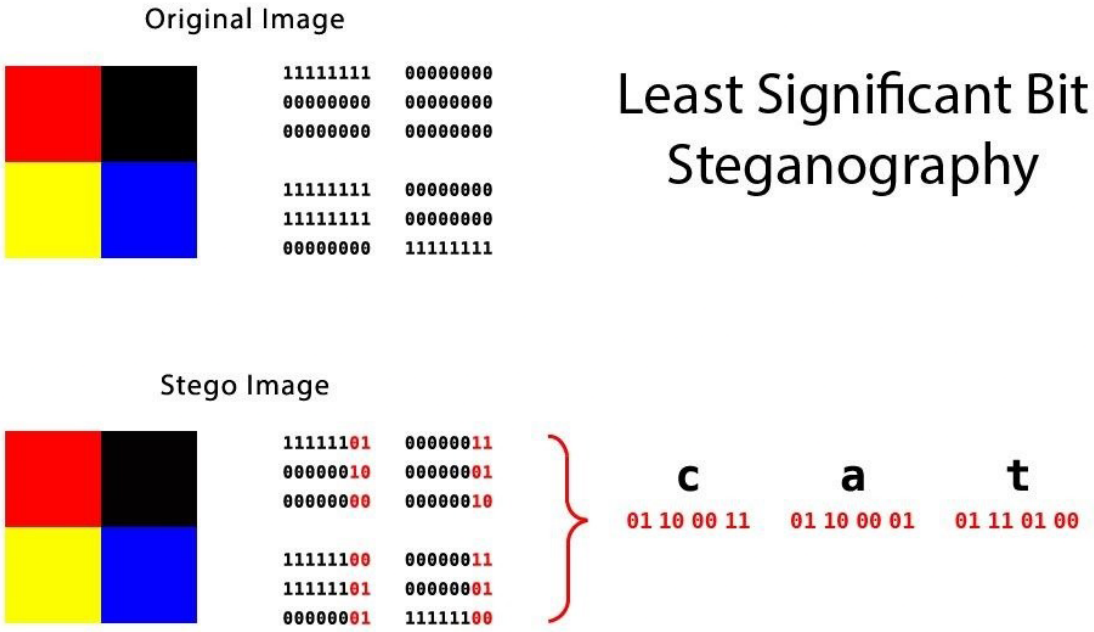
Steganography algorithm

The steganography algorithm should be an algorithm that hides well the data and is not detected easily by the human eye, just like a good watermark does. It is not because we want to "hide" the data strictly speaking but because we do not want to deteriorate the visual artistic creation. By "hiding" the steganography meta-data correctly we will keep the same image quality as the original one.

There exist several steganography algorithms. The most common and the easiest to implement is LSB (Least Significant Bit). In computing, the least significant bit (LSB) is the bit position in a binary integer representing the binary 1st place of the integer. Similarly, the most significant bit (MSB) represents the highest-order place of the binary integer. The LSB is sometimes referred to as the low-order bit or right-most bit, due to the convention in positional notation of writing less significant digits further to the right.



The Data may be concealed by manipulating and storing information in the least significant bits of an image or a sound file. The user may later recover this information by extracting the least significant bits of the manipulated pixels to recover the original message. This allows the storage or transfer of digital information to remain concealed.



Another steganography algorithm is the F5 steganography algorithm which uses the concept of DCT (Discrete Cosine Transform). The Discrete Cosine Transform (DCT) is a finite sequence of data points in the form of a sum of cosine functions that oscillate at different frequencies.

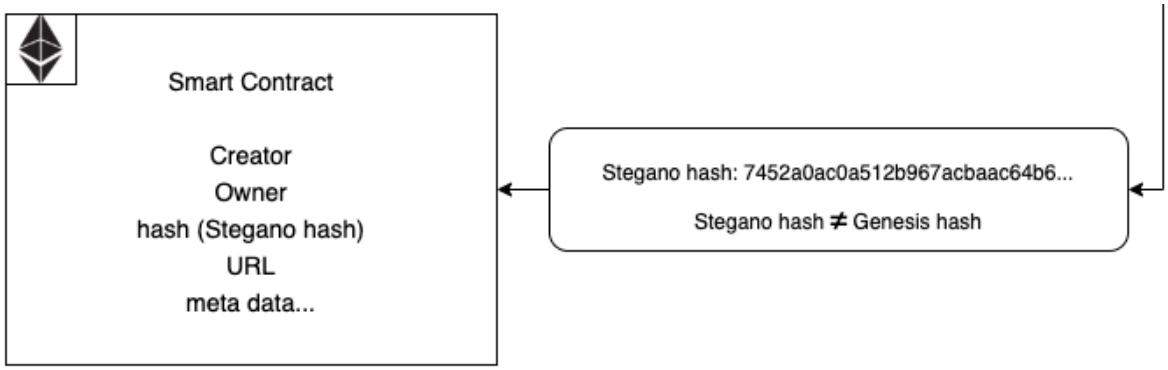
DCT is a widely used conversion method in signal processing and data compression. This is for most digital media including digital images (which can discard small high frequency components such as JPEG and HEIF), digital video (such as MPEG and H.26x), and digital audio (such as Dolby Digital, MP3, AAC), Digital TV (SDTV, HDTV, VOD, etc.), Digital Radio (AAC +, DAB +, etc.), Voice coding (AACLD, Siren, Opus, etc.). The DCT is also important for many other scientific and engineering applications, including Spectral method for digital signal processing, telecommunications equipment, reduction of network bandwidth usage, numerical solution of partial differential equations.

After the steganography

The resulting steganographied file (which we will called stegano file) is the final product that will be stored on IPFS and used as the target file for the NFT smart contract. But we are not done yet. There are some more steps before creating the NFT smart contract.

This stegano file will be hashed by the same hash algorithm used to hash the genesis file. It is better to use the same hash algorithm for the genesis file and the stegano file to not over complicated the process. The resulting hash is called the stegano hash. Note that the genesis hash is a completely different value compared to the stegano hash because the input files used in the hash function are not the same.

The stegano hash is then stored on the NFT smart contract as a storage variable. A variable that is referenced at the minting process as a parameter (like web link of the file) and is not supposed to be updatable. The NFT smart contract should be compatible to existing marketplaces (ERC-721, ERC-777, ERC-1155, etc.).



Along the stegano hash, the URL of the stegano file must be included to retrieve the file. It can be stored on a decentralized place like IPFS or centralized place like S3 bucket or self-hosted. This is a governance file choice, but we recommend using a decentralized storage to maintain the integrity of the file on a permanent storage.

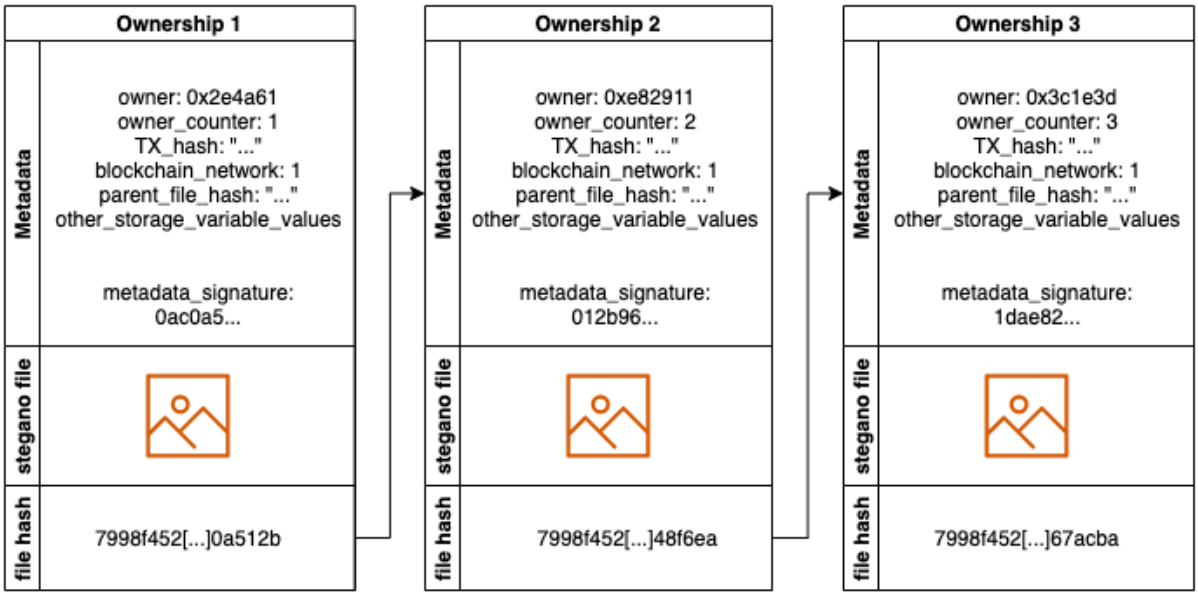
Optionally, the smart contract can include other data such as the genesis hash, genesis meta-data, stegano meta-data, but it is not required. In fact, it is not advised to include it because it can raise the gas cost on the smart contract.

Note that for this process to work as intended to be, only the file owner can be the minter. As we have sometime seen in the NFT ecosystem, some people mint NFTs on behalf of content creators without their authorization and their work. This minting process is not compatible with our concept because the NFT creator needs to sign the genesis meta-data with his private key, and this is an action that cannot be done “on behalf of someone”.

Moreover, the minter needs to be the signatory of the genesis meta-data because the smart contract when deployed is registered in the ledger has been deployed by an entity identifiable by his public key. The same public key used to verify the stegano meta-data containing the message signature.

NFT life cycle

Schema



Legend

Metadata:

Ownership's metadata:

- Owner: Owner's public key of the IP
- Owner counter: current index of entities who successively owned the NFT
- TX hash: Blockchain transaction hash of token ownership transfer
- Parent file hash: Previous Stegano hash (responsible the the "Ownership-Chain")
- Other storage variable values at Block number X (optional)

Metadata signature:
Digital signature of all the above metadata by the owner's private key

Each metadata is different for each ownership iteration (ownership meta-data).
Each metadata is called: metadata 1, metadata 2, ..., metadata N
The Ownership 1 is also called the genesis Ownership (≠ genesis file, which is index 0)
There is no ownership of the genesis file, only of the N stegano file.

Stegano file:

Every N stegano file is created using the genesis file
and the unique iteration of the N stegano meta data.

Stegano file N = Genesis file + stegano metadata N
Stegano metadata N = metadata N + metadata N signature

File hash:

Hash of the N stegano file. Also called stegano hash.
Used in the next ownership iteration metadata (metadata N+1).
The hash must be stored on the smart contract (compulsory).

Explanation

Once minted, a basic ERC721 token for example can be sold and re-sold several times. During the sale, there is an ownership token transfer between the previous owner and the new owner in exchange for some form of currency (fiat, crypto, etc.). In the real world, when an asset is sold, the two stakeholder signs a contract certifying the two steps of the process. First, the seller transfers the token ownership of the targeted asset and second, the buyer certifies the beginning of his ownership of the asset.

With our Watermark NFT contract, this two steps process translates to similar digital actions on the blockchain network. First the current token owner transfers the ownership of the token to the new owner, by using existing smart contract function. For the first steps nothing changes compared to classical NFT norms (e.g., ERC-721).

For the second step there is a lot of changes compared to existing NFT norms. Most importantly, in existing NFT ecosystem there is only one step, the token ownership transfer and that's it. There is no second step of the new token ownership certification. The new owner just "receives" the token without having to sign or input something in the blockchain. But for a Watermark NFT, a new ownership certification must take place by the new owner by "asserting" the event on the blockchain. The new owner needs to perform a new steganography process with new ownership meta-data.

Ownership Meta-data content

The new ownership meta-data contain : the new owner's public key, the owner counter/index incremented by +1, the same blockchain network ID or another ID if the token transfer is cross-chain (inter-chain, sidechains, etc.), a new transaction hash (this time the TX hash not from the mint transaction by from the token ownership transfer transaction) and a new parent file hash (the previous stegano hash and not the genesis hash).

Optional

For the storage variable values, it is still optional and up to the developer to decide to include it or not. But the chosen set of storage variables need to be consistent all along the chain of ownership. If storage variable A, B and C are chosen, at index N the same variables need to be included. The inputted values must be the storage variable values at the block number of the ownership transfer transaction hash specified by TX hash.

Example for the option

For instance, we imagine the token represents an Instagram post with likes and shares as storage variables evolving over time. The developer decides to include the likes storage variable in the steganography. The owner index 7 sells the NFT to a new owner index 8. At the time of the ownership transfer (block number X to be specific), the likes storage variable has a value of 3000. When the new ownership certification by the new owner index 8 takes place, the likes could be at 3005. But the storage variable value that will be included must be 3000 and not 3005, because the ownership transfer appended when the likes storage variable had a value of 3000. Moreover, the specified TX hash (cryptographic hash of the token ownership transfer) refers to a block number where the likes storage variable had a value of 3000.

Including the parent file hash (previous stegano hash) in the meta-data creates a chain of ownership secured by a robust cryptography algorithm that is the hash function. This reproduces the base concept of the blockchain technology by including the hash of the previous element in the current element meta-data. We have a unique chain of ownership for a single token. The File-Ownership-Chain.

The new ownership meta-data must be signed by new token owner using his private key (the same private key to the address owning the said token). The resulting signature certifies the new ownership by the current owner by stating that the owner recognize there has been a token ownership transfer from a previous blockchain address to his current address. “I, Alice, certify that I received a token from this address, with those current ownership meta-data (TX hash, parent file hash, etc.).”

Once the new ownership meta-data signed, both it and its signature must be combined to produce the new stegano meta-data. The stegano meta-data are to be embedded in a new stegano file. The new stegano file is the combination of the genesis file and the new stegano meta-data. For each new owner certification, the steganography process must be done using the genesis file. Once the steganography process done, the file also needs to be stored on a permanent storage.

Finally, the new token owner needs to input the new stegano hash and the new token URL on the smart contract. Once done, the new token ownership has been certified and the step 2 is complete.

Two steps process summary

To summarize, we have two steps in the token ownership transfer. The first step is carried out by the current owner. The token owner’s public key is updated on the smart contract. The second step is carried out by the new owner. The new owner signs the new ownership meta-data, embed it in a new stegano file, store the new stegano file on the permanent storage and input the stegano hash and its location on the smart contract.

If we compare this new process to existing process, the step 1 already in place in existing NFT norms. The new component is the step 2 which includes a compulsory input on the blockchain from new token owner. The two steps process mimics to real life process of an asset ownership transfer. So, compulsory input is more accessible by humans because the real world already imposes this process.

For the new owner to be able to re-sell the token, the new ownership certification must have been carried out, otherwise the token must not be able to be transferred. It is compulsory to include this rule in the smart contract otherwise it would break the chain of ownership. There must a security mechanism preventing the sale of the token so long as the new ownership certification has not been carried out.

File-Ownership-chain

The schema shows a chain of ownership depicting the ownership status of a single token. Each element is a unique ownership instance of the token. This is the same concept as a chain of block (blockchain), hence the name Ownership-Chain.

The Ownership-Chain is a chain of files. Each file is connected to the previous file through the previous file's hash value. Each file contains the hash value of the previous (also called parent) file embedded in its data using steganography. The parent file hash refers to the hash of the previous file in the chain of ownership. The parameter "parent file hash" in the ownership meta-data contains the previous hash and is responsible for the cryptographic chain, just like the parameters "parent hash" inside a block of a blockchain.

Each stegano file has its index which is a positive integer. When dealing with a specific stegano file, we use the term "stegano file index N" (like a mathematic series) or the "N stegano file". By extension we name each stegano hash the stegano hash index N or "N stegano hash". Each ownership meta-data has its index which also a positive integer, also called ownership meta data index N (N ownership meta-data). The ownership meta-data index 1 corresponds to the stegano file index 1 and meta-data index N corresponds to the stegano file index N.

Each stegano file is the combination of the genesis file (genesis file is index 0) and of the ownership meta-data index N (each a unique element). Each N ownership meta-data is signed by the owner index N and generates the stegano meta-data index N (N ownership meta-data + its signature by owner index N = N stegano meta-data, the meta-data to be embedded in the file).

Each element in the Ownership-chain is called ownership index N corresponding to meta-data index N and stegano file index N. The ownership 1 is called the genesis ownership (≠ genesis file, which is index 0). There is no ownership index 0, nor meta-data, nor stegano file index 0. The index 0 is reserved for the genesis file. If we look the Ownership-Chain through a mathematic perspective, it's a pseudo mathematic series with an initial value at index 0 (genesis file) and going through each positive integer by +1 incrementation.

Auditing

There are four steps to audit the NFT and confirming its authenticity.

First retrieve the current stegano file (stegano file from current owner) from its location (usually IPFS) and compute its hash. The resulting hash should be equal to the hash stored in the NFT smart contract storage. If not, the NFT location URL does not correspond to the file.

Second step is to retrieve the ownership meta-data embedded inside the file using the same steganography algorithm used for the ownership certification. The result of the operation should be a JSON document (stegano meta-data). If the meta-data is readable, the audit can continue. If the result is unreadable (not a JSON document), that means the steganography algorithm is not the one used to embed the meta-data or there is no meta-data. Then retrieve the genesis file and use the steganography algorithm to recompute the stegano file. The computed file should have the same hash as the stegano hash in the smart contract.

The genesis file is the “true file” that has been generated by the creator. The stegano meta-data (steganography meta-data) should be in a JSON format, containing tow fields: a message (also as a JSON) and its signature.

Step three. The signature is the combination of the message and the current owner private key. The authenticity of the message needs to be verified using its signature and the so-called owner public key (“so-called” because we don’t know yet if the public key is the true token owner). If the signature is the signature of the message by the token owner’s private key, it confirms that the message was indeed generated by the token owner: “I, Alice certifies I have written this message.”.

The fourth and last step of the audit is to confirm the authenticity of the ownership meta-data embedded in the file. We need to verify owner counter (must be correct owner index), blockchain network, TX hash (must be the transaction hash of the target token ownership transfer on the blockchain network specified by the value “blockchain network”) and parent file hash (must be stegano hash of the previous stegano file in the Ownership-Chain).

If included, verify the value of the storage variable values included. Compare the state the storage variables at the block number the TX hash specified above. The values should be equal, if not the ownership certification is fraudulent. Also, it is a good idea to verify the consistency of the storage variables included. That means to verify that every storage variable embedded in previous stegano files are present (not been drop) and that no other variables have been included.

Business Benefits

By using this new NFT norm, we compensate the existing security in the NFT ecosystem. It first replaces the one-way relationship between the smart contract and the file by a bi-directional relationship between the smart contract and the file with the NFT creator/minter private key as the link. The smart contract contains the file hash (unidirectional link from the file to smart contract) and the file contains the ownership transfer transaction hash (unidirectional link from the smart contract to the file) thus resulting with a bi-directional link between the smart contract and the file.

From a security standpoint, we now have non-repudiation intrinsically embedded in the NFT file thank to steganography and the signed ownership meta-data. Any entity can audit the file to check its authenticity.

We have traceability embedded thanks to the “Ownership-Chain”. It specifies the chain of ownership over time. It is secured by existing cryptographic functions and reproduces the blockchain block linking but with files instead of raw text.

The ownership transfer process mimics the real-world process of assets ownership transfer. It is a two steps process beginning with the transfer of the token by the current owner and ending with the new ownership certification by the new owner.

This new NFT norm can be easily integrated by any NFT ecosystem because it is built upon existing NFT norm and process. It can incentivize several existing platforms to adopt this norm.

Conclusion

To conclude this white paper. This new NFT norm compensates the existing flaws inside the NFT ecosystem. Existing NFT are considered secured because the certificate of ownership (smart contract) is stored in the Blockchain, but it omits the fact the NFT file is stored off-chain, which introduces multiple point of failure and possible breach of trust. Even if the off-chain storage is IPFS.

It introduces the concept of handwriting/brushstroke of the artiste (even for simple content) and is intrinsically embed in the file content. To paraphrase, the file contains the DNA of the creator inside its content.

The content is secured using verified cryptography technologies inside the Blockchain ecosystem, involving security, authenticity, and flexibility. The use of cryptography algorithm to perform the process is customizable. It is recommended to use the same cryptography algorithm used in the target Blockchain for the process, so that there is some form of uniformization, but it is not compulsory.

The steganography algorithm is flexible, but it is recommended to use one that “hides well” the data inside the file, so that the original content is not undermined/deteriorated. Steganography algorithms used by Intelligence agencies are quite popular for watermarking.

The process of auditing the authenticity of the NFT is quick, simple, and transparent. A positive audit result confirms the intrinsic relation between the NFT file, the smart contract and the NFT creator’s private key.

This new NFT norm is an upgrade from existing NFT norms and process. It was designed from the ground up to use the same technologies and cryptography process found in the Blockchain and NFT ecosystem. Moreover, the design was intended to integrate seamlessly with existing Blockchains, NFT marketplaces and Web3 technologies.

This norm should not be considered a definitive solution but as a continuation of upgrades brought to the Blockchain ecosystem. It should and will evolve over time to add more security and authenticity to the NFT ecosystem. Just like the Blockchain technologies are not carved in the stone, technologies and tools need to evolve over time to adapt to new computing capabilities and market demands.

Glossary

Authenticity: the proof an assertion, such as the identity of a computer system user.

Blockchain: A distributed list of transaction records embedded on linked blocks using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a Merkle tree).

Cryptography: The practice and study of techniques for secure communication in the presence of adversarial behavior, it is about constructing and analyzing protocols that prevent third parties or the public from reading private messages various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation.

DCT: The Discrete Cosine Transform is a finite sequence of data points in the form of a sum of cosine functions that oscillate at different frequencies.

Digital signature: a mathematical scheme for verifying the authenticity of digital messages or documents. A valid digital signature, where the prerequisites are satisfied, gives a recipient very strong reason to believe that the message was created by a known sender (authentication), and that the message was not altered in transit (integrity).

Fungibility: In economics fungibility is the property of a good or a commodity whose individual units are essentially interchangeable and each of whose parts is indistinguishable from another part.

Genesis file: Original file generated by the content creator (artist). Used to generate the stegano files.

Genesis hash: Cryptographic hash of the genesis file.

Hash : a mapping of arbitrary size values to fixed-size values. The values are returned by a hash function and are also called hash values, hash codes, or digests. The values are usually used to index a fixed-size table called a hash table.

Integrity (data): the maintenance of, and the assurance of, data accuracy and consistency over its entire life cycle and is a critical aspect to the design, implementation, and usage of any system that stores, processes, or retrieves data.

Intellectual property (IP): a category of property that includes intangible creations of the human intellect. There are many types of intellectual property, and some countries recognize more than others. The most well-known types are copyrights, patents, trademarks, and trade secrets.

IPFS : The Inter Planetary File System (IPFS) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices.

Layer-1: the base layer of the blockchain protocol itself.

Layer-2: third-party integration protocol operating on top of an underlying blockchain protocol that can be used in conjunction with a Layer-1 blockchain.

LSB: the “Least Significant Bit” is the bit position in a binary integer representing the binary 1s place of the integer

Metadata: data that provides information about other data, but not the content of the data.

Mint: the act of creating an NFT

NFT : A non-fungible token is a special type of cryptographic token that represents a digital object such as an image, a video, an audio file, to which is attached a digital identity that is linked to a non-empty set of owners.

NFT Contract Ownership: The ownership of the smart contract containing the token or the token collection. Different from “Token Ownership”.

Off-chain storage: data stores which can be used to store large documents of application artifacts outside of the Blockchain supporting evidence and digital signature.

Owner counter: Incremental counter representing the number of owners over time. Also called owner index.

Ownership-chain: chain of token ownership over time. Copies the concept of the blockchain by using previous element hash and including it in the current content.

Ownership meta-data: Meta-data signed by the token owner at owner counter X. Contains various values representing the past and current of the token.

Peer-to-peer network (P2P): a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.

Parent file hash: Hash of the previous stegano file in the Ownership chain for a given token.

Public-key cryptography: Public-key cryptography, or asymmetric cryptography, is a cryptographic system that uses pairs of keys. Each pair consists of a public key (which may be known to others) and a private key (which may not be known by anyone except the owner).

Signature: (or digital signature) is a mathematical scheme for verifying the authenticity of digital messages or documents. The recipient has very strong reason to believe that the message was created by a known sender (authenticity), and that the message was not altered in transit (integrity).

Smart contract: a computer program or a transaction protocol which is intended to automatically execute, control or document legally relevant events and actions according to the terms of a contract or an agreement.

Stegano file: File with hidden data by a steganography algorithm. Every stegano file is generated by using the genesis file and some ownership meta-data.

Stegano hash: cryptographic hash of the stegano file.

Stegano meta-data: meta-data embedded in the file content using a steganography algorithm. Different from the meta-data.

Steganography: the practice of concealing a message within another message or a physical object.

Storage variable: a single slot in a smart contract that can be queried and altered by calling functions of the code that manages the contract. Also called state variable.

Storage variable value: the value/state of a storage variable at a given block number X.

Token Ownership: Ownership of a unique (fungible or non-fungible) token on a smart contract. Different from “NFT Contract Ownership”.

TX Hash: Blockchain transaction hash. In the white paper, mainly used to describe the Blockchain transaction hash of token ownership transfer.

Watermark: marker covertly embedded in a file content such as audio, video, or image data ensuring data integrity. It is typically used to identify ownership of the copyright or tamper detection of such file.

References

- https://www.wired.co.uk/article/nft-fraud-qinni-art
- https://www.bbc.com/news/technology-58399338
- https://en.wikipedia.org/wiki/Non-fungible_token
- https://ipfs.io/
- https://timdaub.github.io/2021/04/22/nft-sleepminting-beeple-provenance/
- https://www.cryptokitties.co/