On the way to self-contained generative NFT assets immutable lifecycle

Erwan J.
erwan.jpro@gmail.com

2021 November

Abstract

NFTs recently blown-up bringing with it a lot of questionning from the community and blockchain outsiders. In most cases NFTs lacks of the ability to prove ownership, authenticity and uniqueness. We propose a human-readable self-contained NFT asset model with immutable contract ruling its lifecycle in a trustless environment, then we demonstrate the concept with a working example which uses generative 3D blocks built from the asset's hash.

1 Introduction

In a world where blockchain technologies shaked the entire economic system, the community started to sense a larger purpose for it with the democratization of smart-contracts: NFTs were born. NFTs, standing for Non-Fungible Tokens, are unique and non-interchangeable units of data stored on a digital ledger. This token is mostly used to sell supposedly ownership of a wide range of assets ranging from tweets to digital arts.

However the technical details of NFTs implies some flaws that are widely discussed across the community. [1]

On-chain storage is hard, and most of the time impossible because of the large amount of data of assets and most NFTs are just pointing to an url storing the asset [2]. This entierly breaks the decentralized model and put right back a third-party which is subject to data loss and depreciation of services, meaning that an asset could at some point in time be no longer accessible. This forced the community to come up with a decentralized version of assets storage (e.g: IPFS) which is more resilient by design.

Uniqueness of an NFT may also be hard to enforce without the help of some third-party tracker [3]. We've stated before that the token belongs to a digital ledger (e.g: Ethereum) meaning it is considered unique as long as we only takes the ledger it belongs to and the token itself as a reference. At the time of writing, Bitcoin is on the verge of releasing Taproot, a major update allowing for the creation of smart-contracts which will allow the use of NFTs on the Bitcoin

network. Until now, Ethereum was dominating the market of NFTs, but the Taproot update will probably make it harder to track potential duplicates.

Finally, authenticity is most of the time not ensured in any way, meaning that anyone can mint any asset that do not belong to them without effort. They run legal risks by doing this but still, it happens. This is hard for an artist to prove ownership of an asset in the first place without relying on third-parties.

An interesting implementation solving on-chain storage, uniqueness, authenticity and ownership NFTs is done by using generative art algorithms based on a data unique to a token such as a transaction hash [4]. While this satisfies the criterias discussed before, this model lacks of consistency as each smart-contract may contains different code which is hard to verify.

We propose a human-readable self-contained NFT asset model with immutable contract ruling its lifecycle in a trustless environment.

2 Single-hash asset

A single-hash asset self-contains its lifecycle and some data supposed to make it valuable (e.g: a poem) in its plaintext. One issue we face with this model is how to determine that it was honestly created. Nowaday NLP AIs, with the most recent example being GPT-3, are able to easily create entire convincing plaintexts. We can see how easy it would be for an attacker to create ostensibly honest inputs without having to do any work at all. An attacker could also steal other miners inputs and just replace the ownership contract part to reference themselves. Overall the model is too predictable to be safe which is why we opt for a double-hash proof asset method.

3 Double-hash proof asset

A double-hash proof asset works the same as a single-hash asset but adds a unpredictable PoW layer where the hash of the plaintext is taken and maps to an **intermediate proof asset** of the same type for which the miners are given the task to provide a valid proof. Both plaintext and proof hashes are then hashed together to get the final asset hash.

Let H be any hashing function. Let n_x any chosen plaintext. Let $S = \{a_1, \dots a_n\}$, a finite set of assets where $|S| \leq |H|$, we find the corresponding asset hash a as follow:

$$a = H(H(n_x) \| H(proof(H(n_x)))$$

The H-PoW system is formalized as $\forall H(n_x) \in S \;\; \exists H(n_x)_{proofs} = \{p_0, \dots p_n\}$ where p is a valid proof.

4 Proof reuse

If a miner is able to find a plaintext hash of any *intermediate asset proof* which has a known proof, he would be able to use that proof without having to do any additional work. An attacker could build a database of hash proofs that are known to be valid then use a value in the plaintext which can honestly act as a nonce (e.g: an Ethereum address) to bruteforce collisions. However, this comes down to execute a traditional preimage attack which is known to be practically impossible if the hashing function is not broken. Another way to attack when $|S| < |H|$ would be to take the hash of the asset type algorithm output instead of $H(n_x)$ as this gives a higher probability of collisions but at the cost of more computation depending of the underlying algorithm. As the old adage says: *"A chain is only as strong as its weakest link"*, meaning that the lower $|S|$ is, the higher the probabilities of a successful and worth-it attack. Last but not least, depending of the underlying asset type output, an attacker could try to find a plaintext hash which maps to any *proof asset* which is similar enough to another one with a known proof. The flexibility provided by the model allows miners to get creative in order to implement resilient proofs, for instance a miner could make references to the plaintext in his proofs which would then likely betray any miner using the proof with any unrelated plaintext even if the plaintext hash maps to the same *intermediate asset proof hash*. An attacker would then need to preprocess every proofs before using them which might get very hard given their semantical freedom.

5 Unicode scam

$H(proof(n_x))$ is the last step before getting the final asset hash, meaning that if anyone find a way to add arbitrary data without being noticed in $proof(n_x)$, they could effectively bruteforce assets in a very efficient way. Since there is no length limit to a proof an attacker could fill, for instance, as much Unicode zero-width characters as needed in it until finding a valuable asset. However adding too much zero-width characters will inadvertently betray the attacker as the Base64 export will be too large in comparison to the actual readable plaintext. Note that anyone could easily use an online tool designed to spot potential hidden Unicode characters from an input to verify the honesty of any asset.

6 Contracts

If Leonard da Vinci wrote on the Mona Lisa canvas the following sentence: "I must not be sold, at any cost.", would anybody be able to buy it?

When someone buys a work, he buys the original work and its meaning, hence if the work itself states that at any cost, it must not be sold, then we

find ourselves not buying the work as a whole. This paradox is the very basis of contracts.

We define a *Contract* as the portion of plaintext data which rules over the intended asset's lifecycle. Since the *Contract* is part of the asset hash itself, nobody can tamper with it without altering the asset. As soon as any clause of an asset contract is proved to be violated the asset is said to be burnt. A burnt asset has no longer value as long as more that 50% of potential buyers stay honest about the fact that the asset immutable lifecycle must takes precedence over anything else.

Contracts are Turing-complete by design meaning that their applications are limited only by the imagination of those who write them which opens a broad range of possibilites.

7 Uniqueness

We can leverage contracts to provide a way to ensure uniqueness of any asset. We can for example, add a *Contract* ruling over which blockchain the asset can be sold on:

"I may only be minted on Ethereum with a tokenId being my hash"

Note that the syntax is purely a matter of miner's preferences as long as anyone can understand what is stated in the contract.

It is now trivial for anyone to see what the intended lifecycle were from the get-go by looking at the plaintext and if someone tries to sell the asset on the Bitcoin network then the asset is automatically burnt.

8 Ownership

Ownership does not ensure authenticity, the fact that an assets contract set ownership to someone doesn't implies that it was created by this person. On paper, this may sound silly as to why anyone would work for free by giving ownership to someone else, however one of many reasons could be to act as a donation or for any creative purpose. We can define a conditional ownership as follow:

"Whomever mint me with a transaction hash 0xdeed... may own me."

For anyone to mint the asset as a whole, his mint transaction's hash must being with *0xdeed*. However, this asset allows multiple copies as long as the conditions are met.

9 Authenticity

Even if authenticity might not be the priority of a miner since ownership is what matters when coming down to selling an asset, being able to prove that you're the one who created it is necessary.

Most of the time, authenticity is ensured with a public-key cryptographic algorithm such as ECDSA, however this would be too much burden for a miner to add a signature to its plaintext as it would add textual garbage and it might be difficult to agree of signature formats, key formats as much as for the verifier to verify the authenticity of the data.

We propose a **k-timestamp hash proof** technique to allow a miner to ensure authenticity as needed where a miner set an authenticity contract clause by computing $H^k(secret)$ where k is the expiration timestamp in seconds. Let τ be a challenge timestamp, a miner can solve $prove(\tau)$ where $\tau < k$ by providing $H^\tau(secret)$, then the verifier will be able to verify that $H^{k-\tau}(H^\tau(secret)) = H^k(secret)$.

10 Future-proofness

Since NFTs are stored on a blockchain, we can verify the time it was minted. This means that, even if a block uses a proof that is no longer considered safe, as long as the asset was minted at a time where the proof was indeed safe then we should see no impact on the asset's value.

11 Working demonstration

We define our NFT Double-hash asset as a 3D voxel block built with the PRNG function *xoshiro256++* initialized with the asset's hash as state where the hash function is *sha256*. The asset's puzzle will then be another 3D block which the miners will have to use in order to build a valid proof related to it.

12 Algorithms

Our algorithm voluntarily include biasing on voxels colors as without it we're likely to get only uniform colors distribution which would make it too hard to find at least one interesting block.

Rendering the 3D block is done by iterating through $voxelColors$ in z, y, x axes order. Miners can point to any voxel of the block with the formalized syntax $[a.base64(bitmap), ...g.base64(bitmap)]$ where the bitmap is the state of each face voxels where 0 specify unselected and 1 selected.

Algorithm 1 Getting block size and colors

$uniform \leftarrow xoshiro256++(hash)$
$size \leftarrow uniform(1, 16)$
$colorCount \leftarrow uniform(1, 16)$
$voxelCount \leftarrow size > 1?size^3 - (size - 2)^3 : 1$
$colors \leftarrow []$
$voxelColors \leftarrow []$
$colorsBias \leftarrow x = \sum_{n=0}^{voxelCount-1} uniform(1, 100)$
$i \Leftarrow 0$
while $i < colorCount$ **do**
    $colors \leftarrow uniform(0, 2^{24} - 1)$
$i \Leftarrow 0$
while $i < voxelCount$ **do**
    $voxelColors \leftarrow colors[bias(uniform(0, 2^{24} - 1))]$
Return $size\ \ colors\ \ voxelColors$

13 Conclusion

We have proposed a model of self-contained hash assets with immutable lifecycle contracts and built-in PoW mechanism. We started by exploring how single-hash assets could satisfy our needs but demonstrated that the model is too limited to prevent bruteforce attacks. To solve this, we proposed a double-hash proof asset model where a final asset's hash contains a proof for an *intermediate asset hash* based on the hash of a plaintext. We then extended the possibilites of such assets by explaining how contracts can rule over their lifecycles in an immutable way.

14 References

[1] https://news.ycombinator.com/item?id=26444625 [2] conlan, "Cryptographic Hashing and Why Your Tokenized Art Collection is Worthless Without It", https://editorial.superrare.com/2020/07/28/cryptographic-hashing-and-why-your-tokenized-art-collection-is-worthless-without-it/

[3] Lisa Gibbons, "There is a way to protect NFTs from being replicated or lost: This company does just that", https://cointelegraph.com/news/there-is-a-way-to-protect-nfts-from-being-replicated-or-lost-this-company-does-just-that

[4] William M. Peaster, "Talking Ringers with Dmitri Cherniak", https://metaversal.banklesshq.com/p/talk ringers-with-dmitri-cherniak.