2022 4th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS) | 978-1-6654-7158-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/BRAINS55737.2022.9909178

Decentralized NFT-based Evolvable Games

Christos Karapapas, Georgios Syros, Iakovos Pittaras, George C. Polyzos
Mobile Multimedia Laboratory
Department of Informatics, School of Information Sciences and Technology
Athens University of Economics and Business, Greece
{karapapas,syros,pittaras,polyzos}@aueb.gr

***Abstract*—The popularity of blockchain games continues to grow as Non-Fungible Tokens (NFTs) become the center of attention, contributing to the move towards Web3. We leverage the InterPlanetary File System (IPFS) and NFTs, backed by blockchains, to build a flexible, decentralized, and fair baseline system for trading games. Our solution creates a fully decentralized system, where new business models are enabled. In particular, we introduce and support the evolvability of in-game assets, enable their resale with dynamic pricing, depending on their rarity, and automatically provide a cut to the digital artists, without the need for a trusted (third) party. Our system guarantees that the in-game assets will remain online long-term, by orchestrating various decentralized services. Thus, users do not risk losing control over the artefacts or their value, even if the gaming company loses interest, or goes bankrupt. We considered and compared the Ethereum Name Service (ENS) and the InterPlanetary Name System (IPNS) as the naming component of the system and selected ENS for our solution, despite the fact it introduces monetary cost. Finally, we validate our claims and evaluate the feasibility and performance of the proposed system through a proof of concept implementation.**

***Index Terms*—DLTs, NFTs, Ethereum, IPFS, IPNS, ENS, evolvability, decentralization, Web3, mobile games, crypto-games**

I. Introduction

Distributed Ledger Technologies (DLTs) have found application in many aspects of our lives, as they promise secure, trustworthy, and decentralized transactions with the use of cryptographic techniques. Thanks to their fast growing popularity, DLTs have lately caught the eyes of game development industry. Offering solid proof of uniqueness and ownership for assets, they are fertile ground for the development of various types of games.

Gaming models have changed over the years, targeting to keep pace with the evolution of technology, trying to be attractive to users, and at the same time more profitable for the gaming industry. The traditional *Pay-to-Play* model, where users pay upfront for the game and/or the console, has lately been replaced by the *Free-to-Play* model. In the Free-to-Play model, users acquire the game at no cost, but they are incentivized to spend money for in-game assets. The latter has brought to the fore the gaming career path and the eSports industry. The advent of DLTs combined with the trend of gamers to earn an additional income from gaming, has given birth to the *Play-to-Earn* (P2E) model. In the P2E gaming model, not only do users play for free, but they can potentially earn cryptocurrencies. All in-game merchandise they earn playing rely on Non-Fungible Tokens (NFTs) owned by them, and not by the company, which can be sold or exchanged for cryptocurrency.

In 2017, blockchain trading games made their appearance using NFTs and since then, they have been growing in popularity, as well as in market capitalization. From the pioneering Cryptokitties[1], to Axie Infinity[2], the most recent pokemon-like game following the P2E model with over $1,100,000,000[3] total volume. As these players communities grow, more game categories seem to adopt the concept and it is abundantly clear that NFT games are here to stay. Although these games are considered to be Decentralized Applications (DApps), centralization of their media files (relating to or representing the game environment) has been a thorn in the community's side for years. Artwork and metadata are typically stored in the game company's servers, making the need for decentralized storage crucial. To overcome the latter, in a prior work [1], we presented a "fully decentralized" trading game in order to answer arising questions like "Who owns the artwork of the game?," "What happens if the game company loses interest in the game?," "Does the artwork have any value?," etc. As decentralized storage, we proposed the InterPlanetary File System (IPFS) [2]. In this paper, we extend the work presented in [1], while trying to address some of the encountered challenges. We argue that IPFS is inadequate to serve such a purpose, therefore we reckon that a combination of IPFS and its incentive layer, Filecoin, seem to be a perfect fit, as they provide long lasting availability over a P2P network for storing and exchanging data, with multi-platform software and a fast growing community. Furthermore, we claim that the Ethereum Name Service (ENS) provides better performance and introduces better properties, mainly robustness, than the InterPlanetary Name System (IPNS).

Moreover, due to the explosive growth rate of decentralized technologies and the enormous potential of NFTs in decentralized gaming, [3], [4], new business models have recently emerged. In this paper, we realize decentralized gaming that follows the P2E model, through a proposed system, where the gaming company, artists, and users cooperate and interact over decentralized tools, in a tamper-proof

[1]https://www.cryptokitties.co/
[2]https://axieinfinity.com/
[3]https://nomics.com/assets/axs2-axie-infinity/

and auditable manner. Blockchains, IPFS and cryptography are key mechanisms of our system. The contributions of this paper are the following:

- We design and implement a fully decentralized and self sustainable game system. To this end, we achieve orchestration of various heterogeneous decentralized services, incarnating the Web3 paradigm, while we overcome the arising obstacles.
- Our system has many intriguing and novel properties. We realize a decentralized version of "mint-in-sealed-box", a feature borrowed from tangible collectible assets, leveraging threshold cryptography and blockchains. Moreover, we store the metadata file and the artwork of an NFT (in-game asset) on IPFS and Filecoin, which provide tamper-proof file storage and availability through replication. Assets and all the data will remain online in the long-term.
- Our system enables evolvability, through evolvable in-game assets utilizing name resolution services. The avatars and the attributes of an in-game asset (NFT) can be changed over time. We highlight that ENS provides better properties than IPNS and we validate this claim by evaluating and comparing these two name resolution services (from the perspective of our proposed solution).
- Our solution enables new business models, as royalties can be paid to the artists (or any other involved party) at every resale of an in-game asset.

The remainder of this paper is structured as follows: In Section II, we present background information and technology required for our work. In Section III, we unveil the architecture of our proposed system and in Section IV, we present the proof of concept implementation, its evaluation, as well as a comparative analysis of the two naming resolution systems, IPNS and ENS. In Section V, we discuss various aspects of the system and in Section VI, we present and discuss related work in the area. Finally, in Section VII, we provide conclusions, future extensions and potential improvements of our work.

II. Background

A. IPFS Ecosystem

IPFS is a P2P file sharing system, consisting of many novel technologies, aiming to achieve decentralized data storing and low latency file distribution. One of IPFS' main components is the Distributed Hash Table (DHT), a highly scalable coordinator of data lookup among the different nodes. IPFS' DHT is a Kademlia variant. Another significant component is BitSwap, which is a data exchanging protocol. Finally, Merkle DAG, a combination of Merkle Tree and Directed Acyclic Graph (DAG), is used to certify that data exchanged are unique and IPFS does not store any duplicates. Files stored in IPFS can be easily accessed using HTTPs through public gateways. Every file has a unique Content Identifier (CID) and it is named by that.

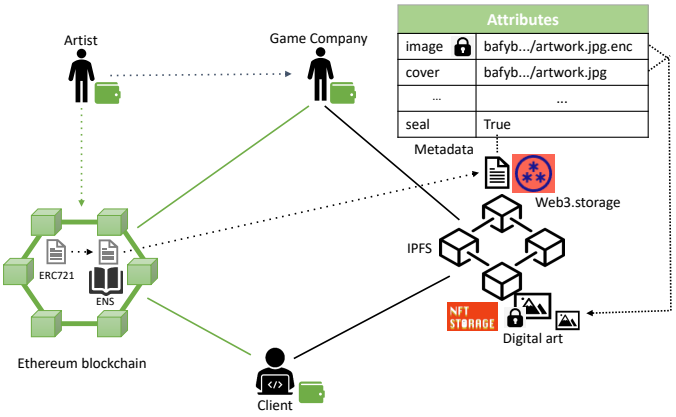

Fig. 1. An overview of the system's architecture.

Although Protocol Labs provides a public gateway [4], lately Web3 related companies, like Pinata, Cloudflare, etc., have been providing their own gateways to the community.

IPNS is the IPFS component responsible for creating immutable addresses pointing to varying data. Every such address is in essence the hash of a public key. For every static address needed, the user should have previously generated an appropriate asymmetric key pair. Once a user wants to update the data pointed by the address, she simply has to publish the new data using the same key. IPNS records are stored on the DHT and also disseminated through it. Moreover, they are versioned, so, when a user queries the DHT for an IPNS address the latest version number is returned. It is noteworthy that IPNS records have a 24 hour lifetime by default, which means that they should be periodically republished to avoid becoming stale.

Finally, *Web3.storage* and *NFT.storage* are open-source services created by Protocol Labs targeting to store general data and NFT related data, respectively, in the Web3 era. Both work in a decentralized manner, leveraging IPFS and Filecoin and are framed by Javascript libraries. It is notable to mention that both services are provided to the community at no cost to the user.

B. Shamir's Secret Sharing

Shamir's Secret Sharing (k,n) [5] is a threshold cryptography scheme, used to secure a secret. Initially, the secret is divided into n fragments, called shares. For the secret to be revealed, at least k of the shares are required. Thus, if the n shares were distributed to n actors, at least k of them must coalite to recover the secret.

III. System Design

In this section, we present the architecture of our proposed system, which is illustrated in Figure 1. Our architecture is composed of the following entities:

- The Ethereum blockchain, the IPFS, and Filecoin infrastructure

[4]https://ipfs.github.io/public-gateway-checker/

- The required smart contracts, one that creates and manages the NFTs and the smart contracts that implements the ENS (registry, resolver, and registrars)
- The gaming company (game administrator/creator) that owns the trading game
- The artists that create the digital arts
- The clients that want to purchase and trade the NFTs and the corresponding digital arts

The actors of the system, namely the *game administrator*, the *artists*, and the *clients*, should own a blockchain wallet in order to be able to interact with the blockchain network. This wallet is also used as an address on the blockchain network and as a secure storage for the acquired NFTs. Moreover, for the encryption part of the digital assets, a symmetric key is generated, split, and shared among the three actors. Finally, the game administrator should own an API key to be able to upload the metadata file and the corresponding digital art on the Web3.storage and the NFT.storage respectively.

From a high level perspective, the entities of our system interact with each other as follows. The artist creates the digital art and sends it to the gaming company. Then, the gaming company creates the NFTs, initializes the corresponding ENS entries, and uploads the encrypted digital art and the appropriate metadata file on the IPFS. Finally, clients can acquire NFTs by paying the defined amount of money (in ethers) on the smart contract. The flow of the system is described in more detail in the following phases.

A. Setup

Initially, the gaming company implements the smart contract that creates and manages the NFTs, based on the ERC-721 token standard, and deploys it on the Ethereum blockchain. We settled on ERC-721, despite the variety of token standards, e.g., ERC-1155, due to its popularity among the blockchain games. Moreover, it is considered as a perfect fit for our system, since we are focusing purely on NFTs, and not on both fungible and non-fungible tokens. The address of the smart contract is considered well-known. Then, the gaming company mints the tokens, receives the media files (e.g., character avatars), which are created by the artists, and it initializes the corresponding ENS entries.

We now describe the flow that happens for each NFT and the corresponding media file. The gaming company generates a symmetric encryption key to encrypt the media file and uploads it on the NFT.storage (the gaming company is considered a trusted entity). Subsequently, the gaming company uploads the metadata file on the Web3.storage. The metadata file contains information about the NFT. Namely, it contains the name and the description of the NFT, the CID of the encrypted media file that corresponds to this NFT, and the CID of the sample cover file. The metadata file contains also a map showing the current owner of the NFT, and a value that shows whether the media file has been downloaded and decrypted. These information are also stored in the smart contract that handles the NFTs.

Upon uploading the media file on the NFT.storage and the metadata file on the Web3.storage, the gaming company modifies the ENS entry to point to the IPFS hash (CID) of the metadata file, in which there is an entry with the hash of the media files on the NFT.storage on IPFS. Finally, it modifies the token URI field of the NFT in the blockchain to point to the corresponding ENS entry.

B. NFT purchase

From this point on, a client can acquire NFTs, either from the gaming company or from another client. For the first case, the client can read the blockchain, by invoking the appropriate function of the token smart contract to find out the available NFTs. Then, if she wants to purchase an NFT, she has to pay the defined amount of money (in ethers) on the smart contract. Subsequently, the token is "transferred" to her account (blockchain wallet) and she is able to see the metadata file of the token stored on IPFS. When a client acquires an NFT, sold for the first time, the gaming company, using Shamir's Secret Sharing (2,3) threshold scheme, splits the decryption key into three parts, and each role of the system, artist, gaming company, and client, gets one.

On the other hand, if a client wants to acquire an NFT that is already owned by another client the following flow occurs. She should come to an agreement with the client that owns the NFT for its price. If they come to an agreement, then she has to pay the agreed amount of money, which is transferred to the smart contract's address. The NFT is "transferred" on the smart contract's address too. The client that sold the NFT, has to send his share of the decryption key to the client that bought the NFT, through the smart contract. Then, an event is generated that eventually is "caught" by the gaming company, which verifies that the key is the actual key and not a fake one. Finally, if everything is as expected, the gaming company sends a transaction on the blockchain, to "transfer" the money on the previous owner' address and the NFT on the new owner's address. Otherwise, the NFT is transferred back to the previous owner and the money is transferred back to the client wanted to acquire the NFT. Due to our design, there is no need for generating a new encryption key or for keeping this specific share secret, since it does not matter if the previous owner of the NFT copies the key or anyone else read the blockchain and acquire the key, as the media file cannot be decrypted with only one share of the key.

C. NFT retaining

In this phase, we assume that a client has already acquired an NFT. If the client wants to download and decrypt the media file, she has to ask the other two parts (artist and gaming company) for their decryption keys. Then, the gaming company checks from the metadata file or the smart contract that the client is indeed the owner of the NFT that she wants to decrypt, and if that is the case,

the gaming company and/or artist send their part of the key to the client, offline and off-chain. After that, the list on the metadata file, as well as the appropriate fields of the smart contract, is updated and shows that the client downloaded and decrypted the media file. The price of the token is adjusted accordingly, based on the status of the media file, e.g., if an asset has not been decrypted by any client yet, its price remains high ("mint in sealed box" feature, see below).

IV. Implementation and Evaluation

A. Implementation

To better illustrate the advantages of our system and to fully quantitative evaluate it, we developed a proof of concept implementation[5]. The developed system is composed of two main parts; the *core service module* and the *smart contracts* that exist on the blockchain network. The *core service module* is a piece of software developed in Node.js. It implements several functions that enable the gaming company to interact with the IPFS-related services, such as the NFT.storage and Web3.storage, as well as with the ENS smart contracts. Moreover, we provide an alternative implementation using IPFS' native name resolution service, IPNS, instead of ENS.

1) Core Service Module: As we have already mentioned above, the *core service module* runs on behalf of the gaming company and implements functions for managing the IPFS and ENS related services. The first function, called `uploadToken`, is used for uploading a token's full sized encrypted artwork, a low quality sample cover, and the token's attributes on IPFS, and in particular on the NFT.storage. Other auxiliary information about the tokens, such as its name, its description, the encrypted artwork file's CID on IPFS, the sample cover file's CID, etc., are included on the metadata file, which is uploaded on the Web3.storage, using the same function. In order for the game administrator to use the two services, two different API keys are required - one for NFT.storage and one for Web3.storage. Each key can be acquired through the each service's website by creating an account. The metadata file also contains information about the owner of the NFT, and whether the digital artwork has been decrypted. This file is encoded using JSON.

The second function of the *core service module* is called `createToken` and is used to create a unique static ENS address for a token. This function is used whenever either a completely new NFT is generated or a token's metadata changes due to a system feature, i.e., evolution and fusion. The function claims a new subdomain, which is used together with the gaming company's domain to identify the NFT. This is illustrated in Figure 2.

Another basic function of our system is the `updateToken` function. This function is used to update the CID of the previously mentioned static ENS address associated with the NFT. Every time that an NFT's metadata is changed, this function is used to update the address to point to the latest version of the metadata file. Finally, the last function is the `decryptToken`, which downloads an NFT's encrypted artwork from the NFT.storage and decrypts it, using two of the stakeholder's shares of the key.

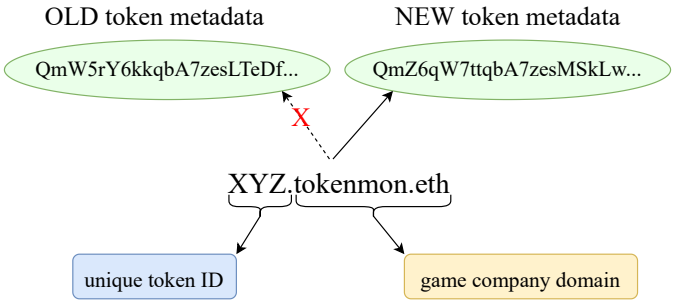

Fig. 2. Diagram illustrating the use of an ENS address in our system.

Two other important functions are implemented separately in a cryptographic module; an `encrypt` and `decrypt` function. Both of them utilize the Advanced Encryption Standard (256-bit AES) via Shamir's threshold secret sharing scheme. The encrypt function is used to encrypt the artist's artwork before going public on the IPFS, via the uploadToken function. A random key is generated and three separate shares are created and distributed to the three entities participating; the gaming company, the artist, and the owner of the NFT. At its current state, the information regarding the keys is stored in a database hosted by the gaming company. Finally, the function results in a fully encrypted file, which can be uploaded on NFT.storage. On the other hand, the decrypt function is used to decrypt the token's artwork using two out of the three secret shares.

1b) Core Service Module using IPNS: As we have already mentioned, every IPNS address is in essence the hash of the public key of a key pair. Thus, for every NFT, a public-private key pair must have been generated in advance. The differences between the two implementations are minimal and are located in the `createToken` and `updateToken` functions. `createToken` generates a new pair of public-private keys that is directly associated with the token's ID. After the creation, the IPFS node automatically saves the pair and an IPNS address is generated via publishing the token's metadata CID by hashing its public key. Finally, a new entry is inserted in the game company's local database including the token's ID, its current metadata CID, and its static IPNS address.

On the other hand, considering that a key pair already exists and that the goal is to update a token, `updateToken` publishes the new metadata CID via the already existing token's public key. At this stage, the static IPNS address points to the newest version of the token's metadata. Then, the function proceeds to update the token's metadata CID field in the local database.

[5]https://github.com/mmlab-aueb/Tokenmon

In order for the implementation to work properly during its early production stages, between random time intervals, a support function iterates the database and republishes the CID for every token ID stored in the database. This is done as a precaution to the IPNS records exceeding their life time problem that is mentioned at the end of the subsection II-A.

2) Blockchain: The other part of our presented system is related to the Ethereum blockchain and the smart contracts. The core smart contract of our system is the NFT smart contract, which is developed using the Solidity programming language and it is deployed on the Rinkeby Ethereum test network (ethers in this test network do not have real value). This smart contract implements the functions that the ERC-721 token standard dictates. In addition to these functions, our smart contract implements three more functions that correspond to the system's actions; `createToken`, `fuseTokens` and `breakSeal`.

The `createToken` function is used to mint an NFT, given a specific URI that corresponds to the ENS address pointing to the metadata file. The `fuseTokens` function is used to merge two NFTs and mint a new one (evolvability feature). Lastly, the `breakSeal` function is used to simulate the seal of a non-digital collectible being broken, thus supporting the "mint in sealed box" feature.

As we have already mentioned, the owner of an NFT is able to choose whether she wants to download and decrypt the digital artwork, namely revealing the NFT's full scale artwork. If such a decision is made, then the break seal function is called, and an event is emitted. The event is eventually "caught" by the core service module software, which updates the metadata file accordingly.

Furthermore, the smart contract implements some (view) functions (introduce zero cost) in order to allow the actors of the system to read and learn auxiliary information about the NFTs. The software that interacts with the blockchain network and the smart contract uses the web3.js JavaScript library. The gaming company can develop its own smart contracts based on the core contract that we implemented, to support more complex game mechanics.

B. Evaluation

One of the contributions of our work is that it provides evolvability. We argue that to achieve this desired feature, a name resolution service is necessary. In particular, we consider two naming systems; IPNS and ENS. In this section, we present the findings from the experiments we conducted, of our twofold implementation, the first using ENS and the second using IPNS.

1) ENS: The usage of blockchain technology is obvious and has a significant impact on the time performance and responsiveness of the system. Furthermore, the invocation of a smart contract function introduces some monetary cost measured in gas units. Various tests have been conducted on the Ethereum Rinkeby test network to estimate the average response times and the cost of the fundamental functions of the core smart contract. The results are are shown in Table I.

Smart Contract Function	Average Response (sec)	Cost (gas)
createToken	27.27	85532
fuseTokens	22.10	99005
breakSeal	17.29	48394
transferFrom	18.35	61523

TABLE I
AVERAGE RESPONSE TIMES AND COSTS IN THE RINKEBY NETWORK.

It must be noted that the presented average response times are significantly exaggerated by sparse long waiting times. By excluding those infrequent spikes, the pure-average waiting times are near the 15 second mark. Other important time measurements have been conducted regarding the ENS-based implementation. One of the most substantial differences between the ENS-based and IPNS-based implementation is accessing the blockchain more frequently to claim a subdomain, update, and retrieve its content hash field. The three functions have also been tested on the Rinkeby test network. The results for these three functions are shown in Table II.

ENS Function	Average Response (sec)
setSubnodeRecord	27.23
setContenthash	14.67
getContenthash	1.69

TABLE II
AVERAGE RSPONSE TIMES OF ENS FUNCTIONS.

2) IPNS: To have a clear picture, in order to compare the two different name resolving systems, we conducted the same measurements in the IPNS network. What we observe in Table III is the time duration from the moment NFT's address is announced over IPNS up to the moment an average user can visit it, through a public gateway. As we have already mentioned, IPNS leverages the DHT. So, the aforementioned times vary, depending on the number of active connections the announcing node keeps. To have a better understanding of the IPNS' robustness and consistency, we conducted the experiments in different versions regarding the number of active connections. Moreover, to have unbiased results every HTTP request is made among a set of different public gateways, and more specifically: (i) https://gateway.ipfs.io, (ii) https://cloudflare-ipfs.com, (iii) https://gateway.pinata.cloud, (iv) https://ipfs.io and finally, (v) https://ipfs.fleek.com.

In Table III, we can see that even using the default settings of IPFS 56% of the total queries timed out, meaning it took more than 5 or 10 minutes, depending on the gateway, to respond. Moreover, we observe that even in the extreme case of over 3000 active connections, which was about the 20% of nodes found online[6], the 27% of

[6]https://trudi.weizenbaum-institut.de/ipfs_crawler.html

requests were not served. From those served, the average response can be also found at Table III. Both the percentage of non-served requests, as well as the average response time cannot be considered negligible and increase the chance that our system will malfunction. Additionally, we should point out that every IPNS record has by default a 24h validity, therefore the game company should reannounce it periodically.

Active Connections	Served	Timed Out	Average Response(sec)
[250 - 600]	28%	72%	23.9
[600 - 900]	44%	56%	20
[3000 - 4000]	73%	27%	15.9

TABLE III
RESPONSE METRICS USING IPNS.

Finally, it is worth mentioning that measuring the exact waiting times of the various combinations of the system functions is at some level meaningless and misleading. For example, the creation, update, and other token mechanisms offered, rely heavily on the user's connection speed. The average mechanism waiting time was observed to be around 1 minute with the create mechanism waiting time being the longest one of them all due to its nature.

Measuring such a quantity can be helpful to determine the user waiting times of the application that is going to be developed by the gaming company. One can easily observe that blockchain waiting times are very noticeable and need to be dealt with in a clever way by the decentralized application in order to ensure the user is immersed. The response times presented in Tables I and II are expected to be longer in the Main Network.

V. DISCUSSION

A. Properties

Our system has many compelling security properties. Initially, it provides availability of metadata. Usually, in games as the one presented there, the gaming company that produced the game hosts the in-game assets locally, on their (centralized) servers, in order to make them available to customers. However, if an outage happens, the servers would become unreachable and the data would become unavailable, causing the whole system to collapse. This is the case also if a gaming company uses IPFS. The gaming company has to provide the media files and the metadata of the in-game assets to the IPFS through their servers. On the other hand, our presented system achieves increased availability due to Filecoin's replication rules, in the sense that data will remain online in any case, even after the game company loses interest or goes bankrupt. Moreover, the proposed system is immune to single points of failure and resilient to Denial of Service (DoS) attacks, as all of its components are decentralized. Furthermore, it is tamper-resistant since all the data are stored on the blockchain and the metadata on IPFS, which is content addressable, i.e., for every chunk of data uploaded a unique CID is generated. Last but not least, thanks to Ethereum's blockchain, it is highly auditable and provides a degree of anonymity known in the literature as pseudonymity.

Every NFT includes a metadata field pointing to the related asset, provided by the ERC-721 standard. For integrity reasons, the aforementioned should not be modified or else NFT loses a part of its value. Dat et al. [6] state that there is a number of marketplaces, among the most popular, which allow tampering with NFT's metadata. Our system considers evolvable in-game assets paired with the corresponding evolvable metadata. To achieve the aforementioned property, we leverage a decentralized name resolution service, i.e., ENS. Thus, when the digital art is updated, there is no need for change in the smart contract, keeping the cost from gas consumption low. The introduction of ENS to the system has offered two crucial advantages: a significantly more user-friendly appearance to the NFT's metadata and lower transaction fees.

In order to update NFT's metadata, its ENS address content field has to be changed. The transaction fee for this action is on average 56854,25 Gwei ($0.10) and it is lower than directly updating NFT's URI via the smart contract, which costs on average 90546,66 Gwei ($0,16)[7]. While the difference might seem insignificant, it greatly increases the game company's long term revenue, considering the amount of transactions that might occur once the game grows in popularity. On the other hand, it is evident that ENS increases the complexity of the system. Despite the initial implications, mentioned in section V-B, the advantages outweigh the disadvantages.

As we have explained in Section III, the owner of the NFT is able to choose whether she wants to reveal the NFT's full scale artwork. This mechanism enables variable token prices, according to their seal state (closed or opened) and could have a significant impact on the revenue generated from the platform both for the business and the artist. Furthermore, the use of decentralized storage combined with the $(2, 3)$ threshold cryptosystem guarantees that even if the company stops supporting the game the owner will be able to fetch and decrypt the artwork.

We have already discussed that the vast majority of blockchain games that utilize NFTs suffer from many shortcomings concerning the actual ownership of the metadata files. For example, a CryptoKitties NFT is a digital, collectible "kitten" built on the Ethereum blockchain. It can be bought and sold using ether and bred to create new cats with different traits. The concept is very simple, but as the creators mention in their whitepaper, NFTs fail to succeed due to *"Provider Dependency; The existence of a digital collectible is dependent upon the existence of the issuing authority. If a digital collectible is created and the creator ceases to exist, the digital collectibles also cease to exist"* [7]. Our solution focuses heavily on the decentralization and preservation of the NFT metadata,

[7]as measured on June 4, 2022.

breaking the dependency of the issuing authority. If the game company that minted the NFTs ceases to exist, the token continues to live on, as its artwork and metadata are safely stored on IPFS via the NFT.storage and Web3.storage services. The rights of ownership are also preserved by the smart contract on the Ethereum blockchain. Other games, such as Sorare [8] and Axie Infinity focus on using their own instance of the blockchain and do not specify the exact technology used to store NFT artworks and metadata. More importantly, the creators do not clarify what would happen in case of a server failure.

Another issue with legacy games is the general lack of NFTs use outside the game environment, also pointed out in [7], as another reason of failure; *"Lack of Function; Physical collectibles are popular because of their intended purpose. Art is a great example: people collect it, it can be worth a lot of money, and it serves a purpose by hanging on the wall as a thing of beauty"*. Such a claim can now be considered outdated, due to the growing adoption of NFTs. The popular social media platform "Twitter" has initiated its own integration of NFTs, as user profile pictures in late January 2022[9]. This is just one of the many examples concerning the future usage of NFTs in various other applications outside of their original uses. However, it is obvious that such a rapid integration by third parties can compromise the integrity of NFTs. For example, if a user displays their NFT as a profile picture, the original full-size artwork is available to the public. A malicious user, e.g., a forger, could steal the original artwork and mint their own token. Suddenly, two identical looking versions of the NFT exist and the forger could contest ownership rights. This problem is successfully addressed in our system, as the full-size original artwork gets uploaded on IPFS encrypted. A smaller, low quality version of the artwork gets uploaded on IPFS unencrypted, as a sample of the original. The above description fits the purpose of a profile picture; a low quality picture being used to represent the account. Of course, forgers could also download the sample used as a profile picture, but it lacks the quality of the original. If they decide to mint a copy, the result would be poor and evident of fraud.

B. Challenges

All these different Web3 components are still in their infancy. So, it is unsurprising that their orchestration would lead to various challenges. Initially, the content hash processing algorithm of ENS automatically converted the CID v1 provided by NFT.storage to CID v0. The latter turned out to cause an error because NFT.storage produces an encoded ("dag-cbor") object, which cannot be translated to v1. To address this issue, we host the metadata file on Web3.storage and the artwork on NFT.storage. In the metadata file, there is a field pointing to the address of the artwork.

We have previously stated that a name resolution service is an essential component of the proposed system, which provides updatability and evolvability to in-game assets. Both ENS and IPNS complete our system and the use of each brings some advantages as well as some disadvantages. So, while in [1], we proposed the use of IPNS, ENS is proven to be more robust than IPNS, in the sense that every request to it is served. On the other hand, as we have seen, even with a large number of active connections, there is a high probability that a request will not be processed in IPNS, causing inconvenience to the users of the system. However, IPNS seems to be slightly faster than ENS under some circumstances, i.e., in case of extremely high number of active connections. Furthermore, it is very important to mention that the usage of IPNS comes at no monetary cost, unlike ENS, in which every registration or update action incurs fees. Finally, we observed that IPNS increases the complexity of using and supporting the system, since, due to the expiration of IPNS registrations, they should be announced regularly on the network. We claim that, overall, the use of ENS is a more rational choice than the choice of IPNS, as there are more points, where it has an advantage, keeping our system more sustainable.

VI. Related Work

The research regarding blockchain and NFT gaming is still in its infancy. Pittaras et al. [8] discuss the feasibility of blockchain-based games. They perform an extensive evaluation of blockchain gaming to showcase the advantages and disadvantages. Min et al. [9] also conduct an in-depth review of blockchain gaming area, categorizing games based on the way they benefit from blockchains; rule transparency, asset ownership, asset reusability and user-generated content. Although they have included various NFT games in the category that leverages blockchains for asset ownership, they do not take into consideration actual ownership of the digital art. Moreover, the reviewed architecture is centralized, storing game assets on a company's file server.

Wang et al. [9] present an overview of the NFT domain. They study the NFT ecosystem from multiple points of view, they conduct a security evaluation and demonstrate opportunities and challenges. The authors present two different protocols for NFT creation. In the first protocol, the artist creates tokens and sells them directly to potential buyers, while in the second, an NFT template is created and NFTs are produced utilizing the aforementioned template. Our system can serve more complex business models, based on decentralization for all interactions. Finally, the authors conclude that NFTs have a great potential in the gaming industry.

On the same wavelength, Rehman et al. [10] conduct a very detailed review of the NFT research area. They

[8]https://sorare.com/

[9]https://twitter.com/twitterblue/status/1484226494708662273?s=21

propose a categorization of NFTs based on their applications to digital art, fashion, collectibles, games (boosting game potential), domain names, virtual worlds and finally sports. Although we believe that the boundaries of the various categories are ambiguous, our system can be seen as a member of more than one of these categories, i.e., collectibles, digital art, and games. Moreover, the authors present multiple challenges NFT technology is called upon to overcome, e.g., security issues, legal issues, etc.

Fowler et al. [3] study the potential of NFTs for game development. They perform an in-depth investigation from different perspectives. They state that using NFTs in gaming can boost players' motivation through consumer-created content in addition to professional artists. Both can benefit by the use of NFTs, due to royalties at every resale in open or in-game markets. Finally, the authors highlight their severe security concerns. Cases in which the company stops hosting the file or the artwork is altered after being sold are some of them. Our scheme overcomes these obstacles, as it considers decentralized storage which is, thanks to IPFS, tamper-resistant and thanks to Filecoin, robust and permanent.

Finally, Muthe et al. [11] highlight the multiple shortcomings arising from game centralization. They argue that centralized storage and computation may lead to privacy leakage and cause high latency. To overcome the aforementioned barriers, they propose an architecture based on proxies for computing and IPFS for storing data. The proposed architecture incentivizes proxies with rewards on Ethereum. Although the design is generic enough, it cannot serve different business models, e.g. outsourcing artwork of the game to artists, who are paid royalties. Moreover, the authors leverage IPFS for data storage without taking into consideration that with the current rules of IPFS, files are hosted (essentially only) by the original uploader; data get disseminated and cached by other nodes only if they become popular. Our proposed system instead supports artist royalties and deals with IPFS limitations utilizing Filecoin and in particular Web3.storage and NFT.storage.

VII. Conclusions and Future Work

We developed and presented here an architecture that aims to create a fully decentralized and self-sustainable system using various Web3 components. More specifically, we leveraged NFTs backed by the Ethereum blockchain in the role of collectibles, as well as the IPFS ecosystem as decentralized file storage. We compared two different name resolution services, namely ENS and IPNS and selected ENS because of its robustness, even though it introduces monetary cost. The proposed system manages to overcome past obstacles related to NFT artwork, not only by giving the NFT owner the full-control of the file, but also the opportunity to keep its value high by choosing not to "open the box."

We realized a proof of concept implementation of our solution to evaluate it under realistic conditions and to compare the two versions of the name resolution service. Furthermore, an API is provided that enables users to easily set up such a system. Two radically different architectures are made available in order for users–typically, gaming companies–to select which one is more appropriate for their specific needs.

Lately, intense discussion is under way regarding the value of an NFT's digital art[10]. Although our system supports adding value to the artwork with the sealed-box emulation, another very interesting direction is the utilization of steganography. We argue that the use of steganographic techniques will be well suited to provide solid proof of authenticity[11] of the digital art. Thus, it is in our immediate plans to experiment with and support these techniques in our solution. Finally, an intriguing extension of our architecture would be a decentralized and company independent way for NFT-related actors (artists, buyers, et al.) to communicate with each other, not only for NFT-related topics, but also through a secure channel to exchange keys.

Acknowledgment

The work reported in this paper has been funded by the Research Center of the Athens University of Economics and Business.

References

[1] C. Karapapas, I. Pittaras, and G. C. Polyzos, "Fully Decentralized Trading Games with Evolvable Characters using NFTs and IPFS," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–2.

[2] J. Benet, "IPFS - content addressed, versioned, P2P file system," *CoRR*, vol. abs/1407.3561, 2014.

[3] A. Fowler and J. Pirker, "Tokenfication-The potential of non-fungible tokens (NFT) for game development," in *Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play*, 2021, pp. 152–157.

[4] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges," 2021.

[5] A. Shamir, "How to Share a Secret," *Commun. ACM*, Nov. 1979.

[6] D. Das, P. Bose, N. Ruaro, C. Kruegel, and G. Vigna, "Understanding Security Issues in the NFT Ecosystem," *arXiv preprint arXiv:2111.08893*, 2021.

[7] D. Labs, "CryptoKitties Whitepaper," in *CryptoKitties: Collectible and Breedable Cats Empowered by Blockchain Technology*, 2018, pp. 4–9. [Online]. Available: https://drive.google.com/file/d/1soo-eAaJHzhw_XhFGMJp3VNcQoM43byS/view

[8] I. Pittaras, N. Fotiou, V. A. Siris, and G. C. Polyzos, "Beacons and Blockchains in the Mobile Gaming Ecosystem: A Feasibility Analysis," *Sensors*, vol. 21, no. 3, 2021.

[9] T. Min, H. Wang, Y. Guo, and W. Cai, "Blockchain games: A survey," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.

[10] W. Rehman, H. e. Zainab, J. Imran, and N. Z. Bawany, "NFTs: Applications and Challenges," in *2021 22nd International Arab Conference on Information Technology (ACIT)*, 2021, pp. 1–7.

[11] K. B. Muthe, K. Sharma, and K. E. N. Sri, "A Blockchain Based Decentralized Computing And NFT Infrastructure For Game Networks," in *2020 Second International Conference on Blockchain Computing and Applications (BCCA)*, 2020, pp. 73–77.

[10] https://www.bbc.com/news/technology-59262326

[11] https://hackernoon.com/hiding-secrets-steganography-in-digital-arts-and-nfts-531z35f0