Crowdfunding Non-fungible Tokens on the Blockchain

Sean Basu[1], Kimaya Basu[1], and Thomas H. Austin[2,3](✉)

[1] Monta Vista High School, Cupertino, CA, USA
[2] 0Chain Corporation, Cupertino, CA, USA
[3] San José State University, San Jose, CA, USA
thomas.austin@sjsu.edu

Abstract. Non-fungible tokens (NFTs) have been used as a way of rewarding content creators. Artists publish their works on the blockchain as NFTs, which they can then sell. The buyer of an NFT then holds ownership of a unique digital asset, which can be resold in much the same way that real-world art collectors might trade paintings.

However, while a deal of effort has been spent on selling works of art on the blockchain, very little attention has been paid to using the blockchain as a means of fundraising to help finance the artist's work in the first place. Additionally, while blockchains like Ethereum are ideal for smaller works of art, additional support is needed when the artwork is larger than is feasible to store on the blockchain.

In this paper, we propose a fundraising mechanism that will help artists to gain financial support for their initiatives, and where the backers can receive a share of the profits in exchange for their support. We discuss our prototype implementation using the SpartanGold framework. We then discuss how this system could be expanded to support large NFTs with the 0Chain blockchain, and describe how we could provide support for ongoing storage of these NFTs.

Keywords: Blockchain · Non-fungible tokens · Crowdfunding · Storage

1 Introduction

As the world moves to an online, digital retail model, there has been a struggle to find ways to reward artists and other content creators for their work.

Non-fungible tokens (NFTs) have been one proposed solution. Artists create their work and then sell it online, with ownership tracked on the blockchain to determine who owns the unique copy of the work of art.

However, while NFTs offer a model for artists to sell their work, they do not intrinsically offer a way for artists to raise funds to help them create their projects in the first place. Additionally, while NFTs offer a good model for storing smaller amounts of content on the blockchain, the cost of storing a larger work of art on the blockchain quickly becomes prohibitive. In this paper, we

S.-Y. Chang et al. (Eds.): SVCC 2021, CCIS 1536, pp. 109–125, 2022.
https://doi.org/10.1007/978-3-030-96057-5_8

highlight how an artist's project to create an NFT can be supported on the blockchain through crowdfunding. In our design, an artist posts a transaction to the blockchain advertising their project. Other clients may then contribute to the project, in exchange gaining a portion of the coins from the initial sale. Once the artist then launches the NFT on the blockchain, the funding campaign is tied to the NFT itself. The artist may then sell the NFT as they see fit, and the artist's backers are compensated automatically. For a successful artist, a history of successful projects can be an excellent form of marketing; backers can see the artist's history on the blockchain and thereby be encouraged to invest in the artist's next fundraising campaign.

To help further understanding of our design, we offer a prototype implementation using the SpartanGold framework [2]. Our implementation is available at https://github.com/taustin/spartan-gold-nft. We then consider the storage of larger NFTs, and discuss our proposed design for the 0Chain blockchain. As part of this discussion, we show how 0Chain's token-locking reward protocol [10] can be used to create an ongoing revenue stream to fund long-term NFT storage.

2 Background

Bitcoin's seminal whitepaper [12] introduced the world to the blockchain as a distributed and decentralized ledger for managing cryptocurrency. Bitcoin also included a primitive scripting language for writing programmable smart contracts. However, due to concerns about denial-of-service attacks, the power of these smart contracts was deliberately restricted.

Namecoin, a fork of Bitcoin, focused on allowing data to be stored directly on the Blockchain. In 2014, Kevin McCoy and Anil Dash used Namecoin to launch what is generally considered the first NFT [6].

Ethereum expanded upon Bitcoin's ideas to create a blockchain that supports a quasi-Turing complete virtual machine [22]. To avoid denial-of-service attacks, Ethereum's virtual machine (EVM) includes a notion of *gas*. Clients pay for their transactions by specifying a gas price; if they run out of gas, the effects of the transaction are rolled back, and the miners keep the ether used to pay for the transaction.

Ethereum's flexibility introduced the world to a wide variety of new applications for the blockchain. One popular use was the creation of ERC-20 tokens [20]. The ERC-20 specification allows a standard way for organizations to issue tokens as a fundraising mechanism. These tokens typically serve as a placeholder for the native coins on a new blockchain; once the new blockchain is launched, clients may exchange these tokens to receive an equivalent amount of native coins on the new blockchain.

While ERC-20 has been an influential design, its focus is on *fungible* tokens. There is no connection between a fungible token and any unique asset. Essentially, ERC-20 tokens act as an additional currency running on the Ethereum blockchain.

To our knowledge, the first example of a fungible token on the Ethereum blockchain was used in the design of *CryptoPunks* [4] in 2017. In this application, users trade unique cartoon characters on the blockchain. Later that same year, *CryptoKitties* was released on the Ethereum blockchain. At its height, CryptoKitties accounted for a quarter of the traffic on Ethereum's blockchain [5]. The popularity of these applications served to both highlight the power of the Ethereum blockchain, and to showcase its limitations in handling the amount of traffic generated by these applications.

The success of these applications lead to the development of two Ethereum Improvement Proposals (EIP): EIP/ERC-721 [7] provides a standard interface for non-fungible tokens; EIP/ERC-165 [15] gives a way to tag an implementation to indicate that it supports a given contract interface.

The ability to create unique tokens on a decentralized, publicly visible blockchain has lead to some initial efforts at using NFTs as a form of inventory management. Regner et al. [14] describe how NFTs can be used as part of an event ticketing system. Westerkamp et al. [21] use NFTs on Ethereum to track inventory in a manufacturing process, where "recipes" dictate how NFTs representing ingredients are consumed to produce new NFTs of the finished good. Stefanović et al. [18] describe the applications for smart contracts in handling land administration systems and real estate transfers, though the authors do not explicitly mention NFTs. Bastiaan et al. [1] describe how NFTs could be useful in real-estate management, including some discussion of early attempted applications of this work for Vermont and Ukraine. Patil [13] develops a NFT-based land registry system using government records for the Washington D.C. area. Salah et al. [16] propose a system for tracking soybeans using the Ethereum blockchain. Kim et al. [8] describe possible applications in the areas of food traceability and describe how these assets can be *tokenized*.

Alternately, NFTs have been seen as a new way to create a market for digital works of art. While CryptoPunk and CryptoKitties can been seen as initial works in this direction, additional challenges remain. Chevet [3] provides an overview of the challenges and benefits in using NFTs to reward artists, arguing that scarcity is the key property that NFTs add to the existing digital art world. Trautman [19] provides a detailed overview of the history of NFTs for virtual art, including extensive discussion of some of the highest-selling NFTs to date.

Muller et al. [11] show how their DeCoCo system can use *fungible* tokens as a mode of rewarding artists, where tokens translate to permission to access some content. While their use case is slightly different than ours, the usage of tokens to track ownership for artistic content bears a similarity to our own design.

3 Crowdfunding NFT Creation

In this section, we highlight how the blockchain can facilitate decentralized fundraising for artist projects, and also tie successful projects to the resulting NFTs. In our discussion, the artist and the backers are both assumed to be clients on the blockchain. There is also a smart contract, the *NFT Smart Contract* (NFTSC), which manages the fundraiser and records the contributions.

We assume that the backers will share the proceeds for the sale of the NFT. However, if the artist wishes to retain all funds, they may specify that when initializing the fundraiser; and non-monetary benefits from the artist must then be managed off-chain. Figure 1 shows a sequence diagram of the process.

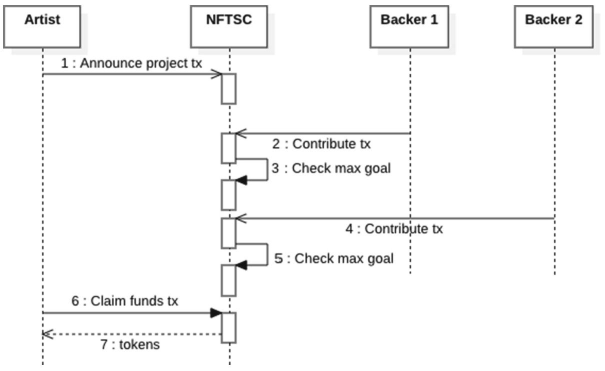

Fig. 1. Crowdfunding sequence diagram

The process for creating a new fundraiser works as follows:

1. The artist posts a transaction the the NFT Smart Contract, specifying:
   - Artist's ID.
   - Project name.
   - Project description.
   - Project ID, chosen by artist. This should be unique for the artist.
   - End date, when the fundraiser will conclude.
   - Minimum funding. If not met by the end date, the fundraiser fails.
   - Maximum funding (optional). If this amount is met or exceeded, the fundraiser ends immediately.
   - Artist share, between 0 and 1. When the NFT is eventually sold, this amount specifies what percentage of the sale goes directly to the artist.
2. Backers contribute to the project, specifying:
   - Artist's ID and the project ID.
   - Amount of tokens to contribute.
   - ID of the backer.
3. The NFT Smart Contract records the contribution. If the maximum funding goal is met, the fundraiser ends.

When the fundraiser ends, the NFT Smart Contract verifies that the funding goal has been met. If so, the contributions are recorded and the funds are transferred to the artist. Otherwise, all funds are returned to the artist. Who initiates the transaction depends on the result. If the fundraiser is successful, it is in the best interest of the artist to write the transaction in order to get access to the raised funds. Otherwise, any of the backers can call the smart contract to reclaim funds from an unsuccessful fundraiser; if there are multiple backers, this situation could result in a waiting game, where each backer hopes one of the others will bear the cost of the transaction. This issue could potentially be addressed by compensating the caller from the contributed funds.

4 Creation of the NFT on the Blockchain

After successful fundraising, the artist can use those resources to create their project. Once completed, they then call the NFT Smart Contract to release the NFT on the blockchain. Initially, the NFT is owned and controlled by the artist, though the backers' shares of the NFT are also recorded.

To create the NFT on the blockchain, the artist must write a transaction calling the NFT Smart Contract with the following information:

- artist ID.
- project ID.
- NFT data.
- NFT content hash (optional).

For smaller NFTs, the entire content might be stored on the blockchain, in which case the content hash is unnecessary. For larger NFTs, the data must be uploaded to its storage location before this transaction is written. Specifying the correct hash is the responsibility of the artist, and the blockchain miners are not expected to validate it. However, the specification of the hash allows others to verify the validity of the content off chain.

Of course, many storage schemes could be used. Section 7 shows how the NFT creation on the blockchain could be coupled with 0Chain's storage system.

Once the transaction has been written, the NFT Smart Contract records the NFT and tracks the ownership information, including details about the backers' shares.

5 Initial Sale of the NFT

With our design, the NFT Smart Contract serves as an escrow service handling the exchange of the NFT for coins on the blockchain. Once the transfer is complete, both the artist and their backers receive their coins, and the buyer receives ownership and control of the NFT. Figure 2 shows a sequence diagram for the steps in the initial sale of the NFT.

To begin this process, the artist writes a transaction calling the NFT Smart Contract. This call should specify:

- The ID of the buyer.
- The purchase price.
- The expiration of the offer.
- The NFT itself. It information should include the project ID.

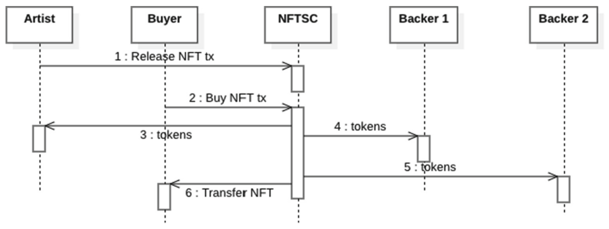

Fig. 2. NFT initial sale sequence diagram

As part of this transaction, the NFT is transferred to the ownership of the NFT Smart Contract. If the offer expires before the buyer has fulfilled the agreement terms, then the artist may call the NFT Smart Contract to reclaim its NFT.

To accept the terms of the agreement, the buyer must write to the NFT Smart Contract, specifying the NFT and transferring enough coins to meet the purchase price. If it does so before the offer expiration, the NFT is transferred to the buyer.

Once the exchange is completed, the proceeds from the sale are transferred to the artist and their backers according to the terms specified in the initial fundraising phase of the project.

6 Implementation

To help further understanding of our design, we implement our system in the SpartanGold blockchain framework. Our implementation is available at https://github.com/taustin/spartan-gold-nft.

6.1 SpartanGold Overview

SpartanGold [2] is a JavaScript framework for simulating different blockchain designs. Its default design is roughly patterned after Bitcoin, with its miners using proof-of-work to validate transactions. However, its design is simpler, and more amenable to being easily extended with alternate designs or configurations. Transactions in SpartanGold include a `data` field, which accepts arbitrary JSON data. As a result, transactions may be extended in a variety of ways without changing the `Transaction` class. However, the logic to correctly interpret any information in the `data` field must be added to the `Block` class, described later in this section. In our design, we add a `type` field to `data`, allowing the `Block` implementation to easily add custom logic for that specific kind of transaction.

Simulations in SpartanGold can be done in a single-threaded mode, communicating through the `FakeNet` module. This approach allows for better demonstrations, with all results posted in a single window. However, the miners may instead be run in separate processes, in which case they can communicate over the network and avoid any "cheats" in the code.

A couple of differences between SpartanGold and Bitcoin should be noted. First, SpartanGold uses an account-based model. In our experience, this model is easier for students to understand than Bitcoin's UTXO model, and it simplifies many cases where we wish to tie some information to a specific account. Second, SpartanGold does not have a built-in scripting language for writing smart contracts.

Instead of using smart contracts, we must extend SpartanGold's `Block` class to handle new types of transactions. The `Block` class not only stores all transactions, but also stores any additional information that should be tracked and keeps track of the rules for validating transactions.

6.2 NFT Basic Operations

For our prototype, we first review how an NFT can be added to the SpartanGold blockchain. While NFTs are often visual works of art, in our example, we use a poet creating a new poem as an NFT. Figure 3 shows the driver for creating and transferring an NFT.

The initial code sets up a blockchain with four clients (`alice`, `storni`, `minnie`, and `mickey`), where two of the clients (`minnie` and `mickey`) are miners, and `storni` is an artist who creates an NFT. The balances for all clients are specified in the genesis block, along with the implementations for transactions and blocks. For this example, we use the standard SpartanGold `Transaction` class, but extend the `NftBlock` class with extra logic for handling NFTs.

After running for 2 s, `storni` invokes her `createNFT` method, where her NFT content is the poem "Hombre pequeñito". The code runs for an additional 3 s before `storni` then transfers the NFT to `alice`. At the 10 s mark, the blockchain terminates, and final balances are displayed. Additionally, the NFTs for `storni` and `alice` are displayed in order to show that the NFT has been successfully transferred.

The output of the program is given below. Some messages have been edited to reduce the output length, but note that `alice` has possession of the NFT at the end of program execution. The balances of the two miners have also increased, each gaining a reward of 25 gold for every block that they have produced.

```
Initial balances:
Alice:  233
Minnie: 500
Mickey: 500
Storni: 500
Mickey: found proof for block 1: 2764
Mickey: found proof for block 2: 4080

... TRIMMED FOR BREVITY ...

Mickey: found proof for block 12: 1268
***CREATING NFT***
Minnie: found proof for block 13: 9984
Mickey: found proof for block 14: 25567

... TRIMMED FOR BREVITY ...

Mickey: found proof for block 23: 54456
```

```
const {Blockchain,Miner,Transaction,FakeNet} = require('spartan-gold');
const NftClient = require('./nft-client.js');
const NftBlock = require('./nft-block.js');

let fakeNet = new FakeNet();

// Clients and miners
let alice = new NftClient({name: "Alice", net: fakeNet});
let minnie = new Miner({name: "Minnie", net: fakeNet});
let mickey = new Miner({name: "Mickey", net: fakeNet});

// Artist creating an NFT
let storni = new NftClient({name: "Alfonsina Storni", net: fakeNet});

// Creating genesis block
let genesis = Blockchain.makeGenesis({
  blockClass: NftBlock,
  transactionClass: Transaction,
  clientBalanceMap: new Map([
    [alice,233], [storni,500], [minnie,500], [mickey,500],
  ]),
});

function showBalances(client) {
  console.log(`Alice:  ${client.lastBlock.balanceOf(alice.address)}`);
  console.log(`Minnie: ${client.lastBlock.balanceOf(minnie.address)}`);
  console.log(`Mickey: ${client.lastBlock.balanceOf(mickey.address)}`);
  console.log(`Storni: ${client.lastBlock.balanceOf(storni.address)}`);
}

console.log("Initial balances:");
showBalances(alice);

fakeNet.register(alice, minnie, mickey, storni);

// Miners start mining.
minnie.initialize(); mickey.initialize();

// Artist creates her NFT.
setTimeout(() => {
  console.log("***CREATING NFT***");
  storni.createNft({
    artistName: storni.name,  title: "Hombre pequeñito",
    content: `
Hombre pequeñito, hombre pequeñito,
Suelta a tu canario que quiere volar...
Yo soy el canario, hombre pequeñito,
déjame saltar.`,
  });
}, 2000);

setTimeout(() => {
  let nftID = storni.getNftIds()[0];
  console.log(`***Transferring NFT ${nftID}***`);
  storni.transferNft(alice.address, nftID);
}, 5000);

// Print out the final balances after it has been running for some time.
setTimeout(() => {
  console.log();
  console.log(`Minnie has a chain of length ${minnie.currentBlock.
      chainLength}:`);
  console.log("Final balances (Alice's perspective):");
  showBalances(alice);

  console.log();
  console.log("Showing NFTs for Storni:");
  storni.showNfts(storni.address);

  console.log();
  console.log("Showing NFTs for Alice:");
  alice.showNfts(alice.address);

  process.exit(0);
}, 10000);
```

Fig. 3. SpartanGold NFT simulation

```
***Transferring NFT fc469b3105a3c89416a...
Minnie: found proof for block 24: 27051

... TRIMMED FOR BREVITY ...

Minnie: found proof for block 45: 66366

Minnie has a chain of length 46:
Final balances (Alice's perspective):
Alice:  233
Minnie: 1125
Mickey: 975
Storni: 500

Showing NFTs for Storni:

Showing NFTs for Alice:

      Alfonsina Storni's "Hombre pequeñito"
      ------------------------------------

Hombre pequeñito, hombre pequeñito,
Suelta a tu canario que quiere volar...
Yo soy el canario, hombre pequeñito,
déjame saltar.
```

```
esCrow.setContract([
  (tx) => tx.from === alice.address &&
          tx.outputs[0].amount === 150 &&
          tx.outputs[0].address === esCrow.address,
  (tx) => tx.from === storni.address &&
          tx.data !== undefined &&
          tx.data.receiver === esCrow.address &&
          tx.data.nftID === nftID
], () => {
  esCrow.postTransaction([{ amount: 150, address: storni.
      address }]);
  esCrow.transferNft(alice.address, nftID);
});
```

Fig. 4. Setting an escrow agreement

6.3 NFT Escrow

Since SpartanGold does not have smart contracts, we must enable another way for an NFT to be transferred between clients. Our solution is to extend the SpartanGold *Client* class to create an *EscrowClient*. An *EscrowClient* can receive gold (SpartanGold's currency) or NFTs just like any other client. However, it can receive a contract of conditions that different parties agree to take through the `setContract` method.[1]

[1] Note that calling this method does not involve transactions on the blockchain, and it does not provide any defenses against abuse; this design simulates what would be done through a smart contract in a blockchain that supported them.

Figure 4 shows an example using the `setContract` method. The `setContract` method take an array of *conditions*, which are callback functions returning true or false. The `EscrowClient` monitors transactions, testing them against these functions. Whenever a condition is satisfied, it is removed from the list of conditions. Once the last condition is met, the `action` callback function is executed, and then the `action` itself is deleted. In the example, the contract monitors the blockchain to watch for `alice` transferring 150 gold to the escrow account and for `storni` to transfer the NFT to the escrow account. Once these actions have been completed, the escrow account posts transactions to transfer to the gold to `storni` and the NFT to `alice`.

6.4 Crowdfunding

For our implementation, the following code snippet shows how a client `storni` advertises a fundraiser, set to expire one minute after the project is posted.

```
storni.createFundraiser({
  projectName: "Un poema de amor",
  projectDescription: "Probablemente pienses que este
     canción es sobre ti, ¿no es así?",
  projectID: "1",
  endDate: Date.now() + 60000,
  minFunding: "20",
  maxFunding: "25",
  artistShare: "0.20",
});
```

The following code shows the `initFundraiser` method of the `NftClient` class. It derives a fundraiser ID from the artist's ID and the artist's choice for project ID, and then stores that fundraiser in the current block.

```
initFundraiser(artistID, projectID, {
  projectName, projectDescription, endDate, maxFunding,
      artistShare,
}) {
  let fundraiserID = this.calcFundraiserID(artistID,
     projectID);
  this.fundraisers.set(fundraiserID, {
    donations: [],
    artistID,
    projectName,
    projectDescription,
    endDate,
    maxFunding,
    artistShare,
  });
}
```

A few changes are then needed in other parts of the code. When the `createNft` method from Sect. 6.2 is invoked, the artist must specify the `projectID` matching the ID she selected during the fundraising piece. Doing so ensures that the contributors receive a share of the proceeds on the initial sale of the NFT. Of course, the artist could neglect to specify the `projectID` and keep the full sale price. However, the artist's fundraising history is on the blockchain, and a history of unfulfilled fundraisers is likely to reduce her success in fundraising again in the future. Of course, she could register additional accounts, but if she is a successful artist, changing her identity would be to her detriment.

In addition, when an NFT is sold initially, the contract with the escrow service must also be changed to reward the backers. The code below shows the modified action that could be registered for the `EscrowClient`. Note the addition of the `project` field with the relevant details of the project.

```
(project) => {
  let payment = 450;
  let artistShare = Math.floor(project.artistShare * payment);
  payment -= artistShare;
  let outputs = [];
  // Giving the artist her cut.
  outputs.push( {amount: artistShare, address: storni.address });
  project.backers.forEach(({ id, amount }) => {
    let reward = Math.floor(payment * amount / project.
        totalDonations);
    outputs.push( {amount: reward, address: id });
  });
  esCrow.postTransaction(outputs);
  esCrow.transferNft(alice.address, nftID);
}
```

7 Storing Large NFTs on the 0Chain Blockchain

For smaller NFTs, it is feasible to store the entire NFT directly on the blockchain. However, as the storage needs increase, it becomes exceedingly expensive (or even prohibitive) to store the data directly on the blockchain. Since our focus is on providing a market for artists, we discuss how our design may be coupled with storage using the 0Chain blockchain.

In this section, we first provide a brief overview of 0Chain's design. Then we show how our system could be integrated into this blockchain.

7.1 0Chain Overview

To understand our design, a few key features of 0Chain's architecture must be understood.

0Chain advertises itself as a high performance decentralized storage network. Its *token-locking reward model* (TLRM) [10] allows for "free" transactions or other services. Instead of paying for service by transferring tokens, clients may temporarily lock their tokens (making them unavailable) in order to generate interest, acting somewhat like a bond where the interest is prepaid. That generated interest may then be given to miners or service providers. Essentially, clients pay in liquidity, but do not permanently lose their tokens.

On the 0Chain blockchain, tokens may be placed in *token pools*. A client can then give signed *markers* to other clients, allowing those other clients to draw funds from the token pools. The combination of token pools and markers is roughly analogous to banking accounts and checks.

0Chain's focus is on creating a marketplace for storage. *Blobbers* provide the storage, curated by the blockchain. Data for the blobbers is erasure coded and encrypted, ensuring that no single blobber is given an inordinate amount of power over the data that it stores. Through the use of proxy re-encryption [17], the client's data can be easily and efficiently re-encrypted for any recipient without revealing it to the blobbers themselves. Clients then pay blobbers in *read markers* and *write markers*, allowing the blobbers to draw on funds from the appropriate token pools (referred to as the read pool and write pool respectively).

7.2 Modifications Needed for Storage

The steps that we outlined in the design of our prototype implementation for the creation and sale of NFTs still apply for NFTs created for the 0Chain blockchain. However, with 0Chain, we can tie the NFTs to storage allocations directly on the blockchain. Note that the data is not stored on the blockchain itself, but the record of payment for storage and the management of the blobbers storing the data is publicly available on the blockchain.

When the artist writes a transaction to the blockchain creating the NFT, they must also specify any needed requirements for storage, such as the amount of data to be stored and the quality of service required. Additionally, they must provide a supply of ZCN (0Chain's native token) to fund the initial storage.

Since the NFT itself is not stored on the blockchain directly, a hash of the content must be stored instead. This hash allows any user accessing the NFT and its off-chain data to verify its authenticity.

When the NFT Smart Contract creates the NFT, it assigns blobbers to store the NFT based on the specifications of the artist. The tokens provided by the artist are divided between the read pool and write pool for the NFT.

Finally, an additional step is needed beyond the process listed in Sect. 4. The artist must upload the erasure coded and (optionally) encrypted data to the blobbers. As part of this interaction, the artist must send signed write markers to each blobber. These markers include a Merkle root [9] of the erasure coded data, thereby serving both as a handshake between the artist and the blobber and as a form of payment. The blobber may write a transaction to redeem these markers on the blockchain, but doing so serves as the blobber's commitment to store the data that matches the Merkle root specified by the client. A challenge protocol probabilistically ensures that the blobber is both storing the data and that it matches this agreed-upon Merkle root. The blobber is rewarded or punished depending on the results of the challenge. For more details on this challenge protocol and the format of the write markers, we refer the interested reader to Merrill et al. [10].

One significant modification is needed to 0Chain's architecture for this design. In the 0Chain ecosystem, allocations of data are permanently tied to a single account. However, with an NFT, we wish to be able to transfer control of the data corresponding to the NFT to the new owner. Adding this ability would be relatively straightforward, and could potentially lead to additional applications.

There is one significant challenge that must be addressed, however. If the data for the NFT is encrypted using proxy re-encryption [17], then the ability to generate re-encryption keys requires the original private key used to encrypt the data. This would require the blobbers to re-encrypt the data when an NFT's ownership changed, who would need to be compensated for their additional work. We will discuss that point in more detail in the next section.

When selling an NFT on 0Chain's network, the ownership of the associated data allocation must also be transferred with it. While this is not a currently supported feature, the change to do so seems fairly minor. When the allocation is transferred to the new owner, the tokens in the corresponding read and write pools remain associated with it. Therefore, the initial cost of storing the NFT will already be handled by the previous owner; handling the ongoing storage is described in the next section.

If the data for the NFT is not encrypted, no other change is needed to the process. Essentially, a storage allocation for an NFT can be handled like any other. Ideally, the allocation should be marked as read-only, thereby guaranteeing to any buyer of an NFT that the content has not changed from the original hash. While this does not guarantee that the data originally updated is correct, an auditing service could be used to review the NFT and provide a stamp of approval on the blockchain.

A more interesting case arises when the data for the NFT is encrypted. 0Chain uses proxy re-encryption. The NFT owner would first erasure code the data into separate stripes given to each storage provider, and then encrypt the stripes using its public key. When providing read access to other parties, the owner would take the receiver's public key; from that public key and the owner's own public/private key pair, the owner generates a re-encryption key. The re-encryption key is sent to the blobbers, who can re-encrypt the data as if it had been originally encrypted with the receiver's public key. The advantage of this approach is that the blobbers do not have access to the original content, but can re-encrypt the data for the receiver on the owner's behalf.

However, when transferring an NFT to a new owner, the data stored must be re-encrypted for the new owner's key pair. Fortunately, this re-encryption can be done entirely on the blobber's side. When the seller writes a transaction to transfer ownership of the NFT, they must include the valid re-encryption key.

The blobbers would need to re-encrypt their storage using the re-encryption key. However, they would need to be compensated for their work, and to ensure that the Merkle root that they have committed to storing matches the data that they are actually storing. As a result, the buyer would need to calculate the Merkle roots for each data chunk and upload matching write markers to all blobbers.

7.3 Funding Storage

For NFTs stored on the blockchain, the initial cost is high, but the NFT owner does not need to pay maintenance costs. Since data on the blockchain is permanent, it will always be available as long as the full blockchain is stored by some subset of the mining network.

In contrast, in 0Chain's ecosystem, storage is an ongoing cost. Clients pay blobbers for a period of storage; when that period ends, the client must negotiate to continue the storage contract, or else let the storage allocation expire. This model allows storage to be done more cheaply, but requires ongoing funds to maintain the NFT.

However, 0Chain's token-locking reward model can be used to create permanent storage. By locking tokens for a set period of time, the client earns additional tokens as a form of pre-paid interest. Those tokens can be spent however the client wishes, and is the basis for 0Chain's "free" transaction model.

In order to create permanent storage for an NFT, the owner needs to create an additional token pool, which we refer to as the *NFT funding pool*. The NFT funding pool may be periodically locked in order to generate an ongoing revenue stream for the NFT's write pool.

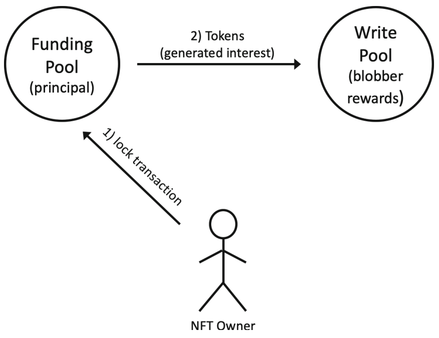

Fig. 5. Fundraising pool

For example, let's assume that the cost of storing an NFT is 20 tokens for 90 days, and that the interest rate for locking tokens is 10% for the same period. The owner can create an NFT funding pool with 200 tokens. By locking the tokens in the NFT funding pool, 20 tokens are minted and added to the write pool for the NFT. When the storage contract duration elapses, the tokens in the NFT funding pool are also unlocked, allowing the owner to relock them, thus continuing funding for the storage. Figure 5 shows a picture of this process.

Of course, this design must consider price fluctuations in the cost of storage and the value of 0Chain tokens, known as *ZCN*. Should the price of ZCN rise compared to the cost of storage, additional rewards are generated for the write pool, and could be used to offset periods where the price drops.

On the other hand, if the cost of storage drops below the amount of tokens that the NFT funding pool can generate, then the same blobbers will be unwilling to provide storage. However, other blobbers with lower quality of storage could be used as backup storage providers. We introduce the notion of *archival blobbers*; these blobbers would provide cheap storage, but with extremely low read rates. In their role, they could help NFTs to weather sudden, unexpected rises in the cost of storage relative to the value of ZCN.

Some client must initialize the transaction and pay the cost of that transaction. With 0Chain, certain types of transactions are designated zero-cost; the re-locking of NFT funding pools could be added to this list. Alternately, a portion of the minted tokens could be given to the client to compensate them for the transaction fee, or even to provide a small reward for calling the transaction.

The ability of the token-locking reward protocol to create a steady revenue stream seems likely to be useful in a number of other areas. Whenever an ongoing service needs to be funded, this design provides a model of how that funding could be achieved.

8 Conclusion and Future Work

In this paper, we have proposed a system for helping artists to produce NFTs on the blockchain. Our crowdsourcing mechanism both helps artists to create their new projects and more easily reward their backers with a share of the proceeds. We also show how the 0Chain blockchain could be leveraged to store large NFTs and how a revenue stream could be created to offset the cost of that storage.

In future work, we intend to explore how these NFTs could be transferred across blockchains. Additionally, we intend to expand our prototype to further explore the challenges of NFTs.

References

1. Real estate use cases for blockchain technology. Enterprise Ethereum Alliance - Real Estate Special Interest Group, vol. 1 (2019)
2. Austin, T.H.: SpartanGold: a blockchain for education, experimentation, and rapid prototyping. In: Park, Y., Jadav, D., Austin, T. (eds.) SVCC 2020. CCIS, vol. 1383, pp. 117–133. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72725-3_9
3. Chevet, S.: Land registry on blockchain. Blockchain Technology and Non-Fungible Tokens: Reshaping Value Chains in Creative Industries. Master's thesis, Paris, France (2018)

4. Cryptokitties, cryptopunks and the birth of a cottage industry. Financial Times (2018)
5. Cryptokitties key information. https://www.cryptokitties.co/technical-details. Accessed April 2021
6. Dash, A.: NFTs Weren't Supposed to End Like This. The Atlantic, Washington (2021)
7. Entriken, A.W., Shirley, D., Evans, J., Sachs, N.: EIP-721: ERC-721 non-fungible token standard (2018). https://eips.ethereum.org/EIPS/eip-721
8. Kim, M., Hilton, B., Burks, Z., Reyes, J.: Integrating blockchain, smart contract-tokens, and IoT to design a food traceability solution. In: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 335–340 (2018). https://doi.org/10.1109/IEMCON.2018.8615007
9. Merkle, R.C.: Protocols for public key cryptosystems. In: 1980 IEEE Symposium on Security and Privacy, p. 122 (1980)
10. Merrill, P., Austin, T.H., Thakker, J., Park, Y., Rietz, J.: Lock and load: a model for free blockchain transactions through token locking. In: IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON). IEEE (2019)
11. Müller, M., Janczura, J.A., Ruppel, P.: DeCoCo: blockchain-based decentralized compensation of digital content purchases. In: 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services, BRAINS 2020, Paris, France, 28–30 September 2020, pp. 152–159. IEEE (2020). https://doi.org/10.1109/BRAINS49436.2020.9223299
12. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf. Accessed 1 April 2021
13. Patil, M.: Land registry on blockchain. Master's thesis, San José State University, San Jose, CA, USA (2020)
14. Regner, F., Urbach, N., Schweizer, A.: NFTs in practice - non-fungible tokens as core component of a blockchain-based event ticketing application. In: Krcmar, H., Fedorowicz, J., Boh, W.F., Leimeister, J.M., Wattal, S. (eds.) Proceedings of the 40th International Conference on Information Systems, ICIS 2019, Munich, Germany, 15–18 December 2019. Association for Information Systems (2019). https://aisel.aisnet.org/icis2019/blockchain_fintech/blockchain_fintech/1
15. Reitwießner, C., Johnson, N., Vogelsteller, F., Baylina, J., Feldmeier, K., Entriken, W.: EIP-165: ERC-165 standard interface detection (2018). https://eips.ethereum.org/EIPS/eip-165
16. Salah, K., Nizamuddin, N., Jayaraman, R., Omar, M.: Blockchain-based soybean traceability in agricultural supply chain. IEEE Access **7**, 73295–73305 (2019). https://doi.org/10.1109/ACCESS.2019.2918000
17. Selvi, S.S.D., Paul, A., Dirisala, S., Basu, S., Rangan, C.P.: Sharing of encrypted files in blockchain made simpler. In: Pardalos, P., Kotsireas, I., Guo, Y., Knottenbelt, W. (eds.) Mathematical Research for Blockchain Economy. SPBE, pp. 45–60. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-37110-4_4
18. Stefanović, M., Ristić, S., Stefanović, D., Bojkić, M., Pržulj, D.: Possible applications of smart contracts in land administration. In: 2018 26th Telecommunications Forum (TELFOR), pp. 420–425 (2018). https://doi.org/10.1109/TELFOR.2018.8611872
19. Trautman, L.J.: Virtual art and non-fungible tokens (2021). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3814087
20. Vogelsteller, F., Buterin, V.: EIP-20: ERC-20 token standard (2015). https://eips.ethereum.org/EIPS/eip-20

21. Westerkamp, M., Victor, F., Küpper, A.: Blockchain-based supply chain traceability: token recipes model manufacturing processes. In: IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), iThings/GreenCom/CPSCom/SmartData 2018, Halifax, NS, Canada, 30 July–3 August 2018, pp. 1595–1602. IEEE (2018). https://doi.org/10.1109/Cybermatics_2018.2018.00267
22. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger (2014). https://gavwood.com/paper.pdf