A Technical Deep Dive Into and Implementation of Non-Fungible Tokens in a Practical Setting

Julia Martin, Carrie Hay Kellar
jm5251, cah2251

December 2021

1 Introduction

1.1 Background

Since late 2017, Non-Fungible Tokens, or NFTs, have gained widespread popularity amongst the blockchain community, providing a new kind of blockchain-based token that has a unique value pertaining to the individual token data. In contrast to fungible tokens (i.e. cryptocurrencies such as Bitcoin and Ether), NFTs do not have a standardized value, nor can they be divided or merged – hence, not fungible. They represent a whole entity/asset, rather than a unit that can be combined or broken into multiple pieces (so long as the total value stays the same). Essentially, they were created as a way to tokenize distinct, collectible items, and have since been adapted to encapsulate many kinds of digital tokens, from tweets to tickets. Offering NFTs on blockchains allows them to be a part of standard public/private key transactions, in which the purchasing party's private crypto key constitutes a proof of ownership of the original token, which is transferred to their digital wallet, and the public key shows a certificate of authenticity from the creator. However, this does not prevent copies of the token from being shared and distributed. The combination of the two is used to derive the tokens value.

When a game called *CryptoKitties* rose to prominence in 2017, in which players could purchase and breed virtual cats, game developers everywhere began to adopt the concept of NFTs as a way to allow players to collect digital, in-game items [1]. This asset tokenization enabled the items to be easily transferred between players even other games via specialized NFT marketplaces, and the realized potential grew from there. Ethereum became the first blockchain to embrace the idea, and it introduced a standard smart contract *ERC-721* to support them in 2018. Ethereum is still the most widely used blockchain on which NFTs are stored; however, many others have started to implement their own versions of NFTs (such as Bitcoin, Cardano, Solana, etc). Since then, they have become ubiquitous in popular culture and brand competition, and the market has experienced exponential growth, with its current market cap nearing $5 Billion [2].

1.2 Motivation

The purpose of this paper is to provide a technical deep dive into non-fungible tokens. We want to look beyond the basics and the easily digestible details and discover how NFTs work on different blockchains and how they utilize blockchain technology. While preparing this comprehensive guide, we will examine a few different aspects of NFTs. After starting with an explanation of how they

work and are created on blockchains in general, we will take a more specific look at their initial implementation on Ethereum. We will then compare and contrast the Ethereum NFT standard with that of Solana. The reason we chose Solana here is because it is growing extremely quickly, and has the potential to be Ethereum's largest competitor in this market [3]. These two blockchains will be compared from the viewpoint of using dynamic data and from the viewpoint of the different formats that NFTs take.

Beyond our theoretical explanation of the concept, and in order to obtain a more applied and in-depth grasp of the implementation, we will actually code and mint our own NFT onto the blockchain with dynamic data. Using Rust, we connect to Solana and create an account to mint and to which we send the NFT. From here we disable minting of additional tokens identical to this one. Once we have a unique token on the Solana blockchain, all that is left to do is link it to an asset (such as a piece of art), and it is ready to be transacted on the blockchain as an NFT.

The final portion of this paper is devoted to the limitations and issues with NFTs, including (but not limited to) those regarding blockchain transactions themselves. Essentially speaking, what are the challenges of practical usages of NFTs, and do the benefits outweigh those challenges? Along the same lines, we can examine what the future holds for this technology and what it's possible uses will be. This leads into an exploration of the cultural and economic impact of NFTs, and how they contribute to decentralized finance as a whole. This paper assumes that the reader has a fair understanding of how blockchain technologies work and the underlying theory behind cryptographically secure transactions, although we will go over a very short smart contracts review in the next section.

2 Technological Deep Dive

2.1 General Information

As we laid out above, we know that a non-fungible token is used to represent any unique asset on a blockchain. One main difference between fungible and non-fungible tokens (apart from the obvious substitution/fungibility differences), is that non-fungible tokens each have specific identities (which are included in transaction metadata). They also have the ability to enable special ownership functions, which fungible tokens do not have. On the other hand, similar to fungible tokens, they are powered by smart contracts. These contracts are distinct to the individual chain on which the token is hosted, and are written in different languages depending on that chain (i.e. Solidity for Ethereum, Rust for Solana). Given that anyone with a working knowledge of a network's contract language can write and deploy a smart contract, minting NFTs on smart contracts gives the creator increased control over the transaction process, provided that they are able to pay for deploying the contract. Along the same lines, given that the contract is replicated across the nodes of the chain, parties initiating a transaction on the contract can indicate certain parameters that execute specific steps within the contract [4]. Smart contracts are well suited to handle NFT trading not only because they can handle standard transaction processes such as the payment of funds, but they can also ensure the transaction security by imposing financial penalties on unmet conditions. This contributes to the decentralization of the trading game by eliminating the need for human intervention once a smart contract has been deployed.

Using the deployed smart contract, an NFT can be minted and published onto the chain as follows. The contract function that we use to mint an NFT itself takes in the NFT's metadata

describing what the asset actually is. The creator uses their private/public keys and sets up a minting transaction which includes the following info:

1. Transaction origin (from) - this is the creators public key
2. Transaction destination (to) - this is the contract address with which we are interacting and to which we are sending the transaction
3. Account nonce - keeps track of the number of transactions sent from the creators public key
   (a) This is used for security purposes; i.e. preventing replay attacks
4. Gas - estimated cost to complete the transaction

The creator then signs the transaction with their private key and sends it to the contract [5]. Then a new certificate is generated for the new owner, which is used for proof of authenticity.

As we've seen, in order for a blockchain to support NFTs natively it must have smart contract capabilities in its Layer 1 protocols. This specific infrastructure does a lot to limit the protocols which can effectively mint NFTs. For example, Bitcoin does not have the smart contract capabilities that, say, Ethereum does, so it cannot natively run the code that is necessary (although it is adapting an approach utilizing Stacks and rolling up all transactions and settling them on Bitcoins L1). It also helps to keep transactions secure via cryptographically secure core protocols. Currently, Ethereum is the most commonly used standard for creating and housing NFTs, popularly utilizing smart contracts to maintain flexibility of the blockchains abilities. However, the space is rapidly changing, and many other protocols are bursting onto the scene. For example, blockchain platform Flow is an alternative that allows developers to build blockchain applications directly, many of which involve collectibles and games in the NFT space. Another interesting one is Concordium, an L1 Blockchain which utilizes PoS with an ID layer at the protocol level. Solana is another quickly growing platform that offers lower price and faster usage than Ethereum. In the following two sections, we will lay out the details of Ethereum and Solana NFT standards, how they work, and how they compare to each other. We have chosen Solana specifically to dive into because it has grown 4800% since September 2020 and has the potential to be Ethereum's main competitor.

2.2 Ethereum

Created in 2018, EIP-721 was Ethereum's first proposed standard for an implementation of Non-Fungible Tokens, providing a blueprint from which other platforms could grow. It allowed for different tokens to have different values generated from the same smart contract (as outlined above), due to their ids. In Ethereum, all NFTs have a *uint256* variable called *tokenId* such that for any contract the pair (*contract address, uint256 tokenId*) is globally unique [6]. The 721 proposal also outlined a standard API using Solidity that would be used within smart contracts for NFTs, which provided the basic functionality needed to transfer and track them. This includes functions such as the following[7]:

```
pragma solidity ^0.4.20;

interface ERC721 {
    //emits when ownership of any NFT changes by mechanism
    event Transfer(address indexed _from, address indexed _to, uint256 indexed
    _tokenId);

    //emits when approved address for NFT is changed (when Transfer event happens,
```

```
    //approved address for NFT is set to none)
    event Approval(address indexed _owner, address indexed _approved, uint256
    indexed _tokenId);

    //emits when operator can manage all NFTs of owner
    event ApprovedForAll(address indexed _owner, address indexed _operator, bool
    _approved);

    function balanceOf(address _owner) external view returns (uint256);

    function ownerOf(uint256 _tokenId) external view returns (address);

    function transferFrom(address _from, address_to, uint256 _tokenId) external
    payable;
}
```

Listing 1: EIP-721 API Example

Ethereum also in this standard declared the token receiver interface which must implement the wallet interface to receive safe transfers

```
interface ERC721TokenReceiver {
    function onERC721Received(address _operator, address _from, uint256 _tokenId,
    bytes _data) external returns(bytes4);
}
```

Listing 2: EIP-721 Token Receiver

And Ethereum includes an NFT specific metadata extension so that the smart contract can provide details about the NFT asset.

```
interface ERC721Metadata {
    function name() external view returns (string _name);

    function symbol() external view returns (string _symbol);

    function tokenURI(uint256 _tokenId) external view returns (string);
}
```

Listing 3: EIP-721 Metadata Extension

The Ethereum NFT Metadata which is referenced above can be seen as

```
{
    "title": "Asset Metadata",
    "type: "object",
    "properties": {
        "name": {
            "type: "string",
            "description": "Identifies asset which this NFT represents"
        },
        "description": {
            "type": "string",
            "description": "Describes asset which this NFT represents"
        },
        "image": {
            "type": "string",
            "description": "URI pointing to resource with image representing the
    asset which this NFT represents"
        }
    }
}
```

Listing 4: EIP-721 NFT Metadata JSON

We note here that a contract complying with the API laid out above currently has certain caveats (laid out in the Ethereum docs) due to Solidity not having expressive enough interface grammar [7]. However, the EIP-721 standardized interface that Ethereum created above makes the NFT ecosystem much easier to manage given that it allows for cross-functional asset management. This is why the creation of EIP-721 is widely credited with the beginning of the NFT boom - it made it possible to create a market out of NFTs.

The implementation and deployment of this API on a contract is left up to the developer, as is the creation and desctruction of NFTs, although the standard creation of NFTs on the Ethereum chain works as we laid out in the above section. Instinctively, transfers may only be initiated by parties that are currently associated with the NFT:

- The owner
- The approved address
- An authorized operator of the current owner

However, although the transfer can be initiated by any of those three, the authorized operator only may set the approved address for the NFT. One of the most interesting and applicable aspects of the transfer (and accept) functions in this API is that they are flexible based on individual implementation. For example, specific deployed contracts using these functions can throw errors and disallow transfers in other ways not laid out in the EIP-721 documentation, such as blocking specific addresses from receiving NFTs. The destruction ("burning") of NFTs is also left up to the discretion of the implementation, although we often see it done by transferring the asset to an address that makes it impossible to recover any asset. In other protocols, this is often seen as the zero address; however, the Ethereum standard prohibits assigning NFTs to the zero address, and the function will throw for queries about the zero address, so a new contract would have to outline specific proprietary burning approaches on an asset-by-asset basis.

Ethereum has run into some issues in the past when scaling the 721 standard, particularly exacerbated by the fact that the platform on average only supports about 30 transactions per second. One famous example is that of CryptoKitties in 2017. The application became extremely popular, and it was taking up a significant amount of available space for transactions on the blockchain. As it continued to grow, the price of quickly executing transactions rose, and Ethereum experienced a serious latency issue [8]. Likely, this was due to the use of for/while loops in the code, so as the application grew it was not able to scale and gas costs rose without bound. However, since then Ethereum has deployed a new contract which instantiates and tracks 2^{128} different deeds and is queryable from the blockchain, meaning that every function takes less gas than querying from the Ethereum Name Service [7].

Ethereum remains the standard in NFT minting today. It has been around the longest, and it is largely decentralized, which helps prevent disruption or abuse by creators. In addition to that, the use of public/private keys as certificate of authenticity/proof of ownership allows a fairly high security confidence. With the rise of Solana, however, the standard may begin to change.

2.3 Solana

Developing on Solana is like writing code in C, where as developing on Ethereum is like writing in python.

All of the abstractions in Ethereum are taken away in Solana, such that there are only a few key concepts, however putting them all together correctly takes a deep understanding of the architecture. The main three concepts are addresses, accounts and instruction data. Addresses specify where accounts live on-chain, and instruction data specifies the actions of those accounts.

Every account in Solana lives at a certain address, which is a 32 byte string of alphanumeric characters. It is basically a pointer to an account that occupies the address. There are multiple types of addresses in Solana, namely key pair, program derived and seed. We will only be discussing key pair addresses which are used for traditional cryptographic signatures on transactions and program derived addresses when discussing metadata. Each key pair is found using a mathematical curve, such that each public key has a unique corresponding private key which cannot be duplicated since they are deterministically found.

The full commented code is also included in in the appendix, it is written in javascript and calls the endpoints. However, it is simpler to understand through the command line.

In order to make an NFT in Solana, one first has to create a wallet. There are many types of wallet infrastructures in Solana, i.e. Phanton, SolFlare, etc. In fact you can make wallet through the command line, as it is just the generation of a private key/public key pair or an address that is used to sign to allocate an account to an address.

```
$ ~ mkdir ~/fobwallet
$ ~ solana-keygen new --outfile ~/fobwallet/my-keypair.json
Generating a new keypair

For added security, enter a BIP39 passphrase

NOTE! This passphrase improves security of the recovery seed phrase NOT the
keypair file itself, which is stored as insecure plain text

BIP39 Passphrase (empty for none):
Enter same passphrase again:

Wrote new keypair to /Users/carriekellar/fobwallet/my-keypair.json
==========================================================================
pubkey: 7frLwuWeFpvF643ypXCWiuddjMKgzuYio6CsAhxZFr5d
==========================================================================
Save this seed phrase and your BIP39 passphrase to recover your new keypair:
rural oil resist client nasty primary security garage reject deal green upper
==========================================================================
```

To find what our private key is:

```
$ ~ cat ~/fobwallet/my-keypair.json
[228,187,35,234,166,94,81,236,32,21,23,221,203,248,247,43,91,26,18,101,76,241,126,
184,148,175,184,130,207,61,10,15,99,25,252,82,184,169,96,194,207,27,86,96,7,233,
42,173,81,68,21,138,104,220,131,118,120,249,56,109,49,172,215,0]
```

Verification that this is the corresponding secret key to the private key

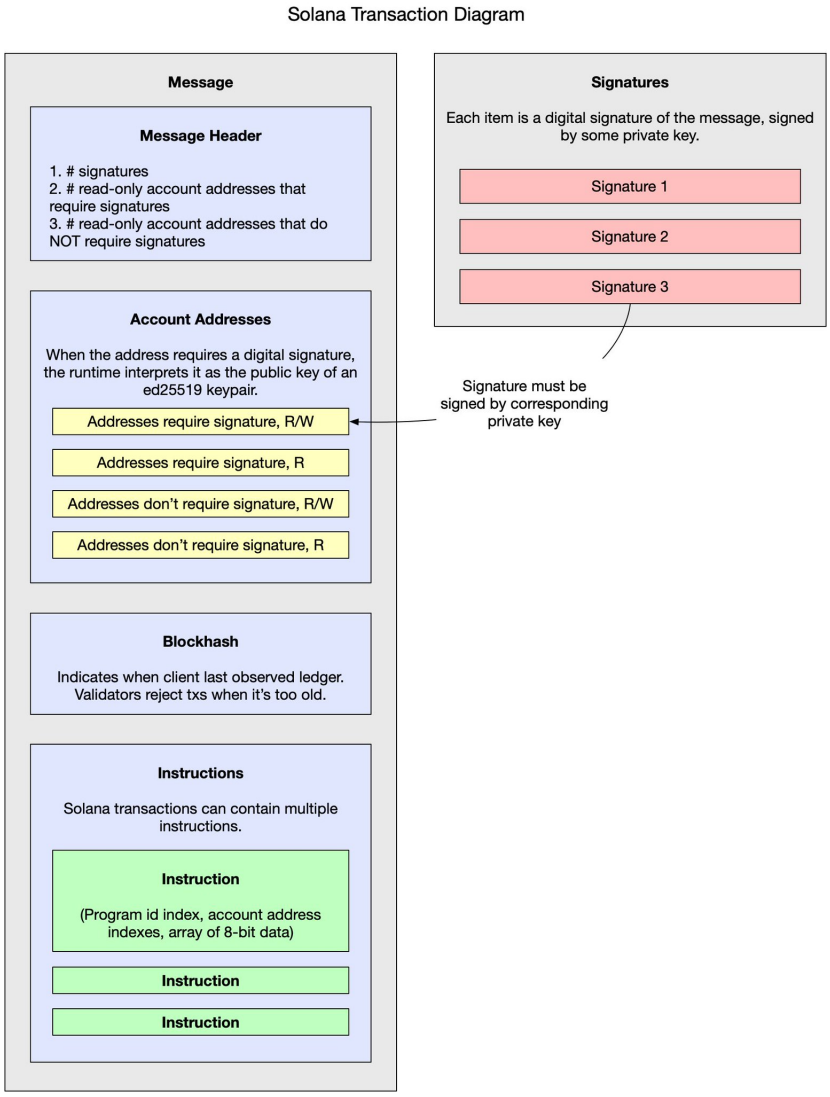
```
$ ~ solana-keygen verify 7frLwuWeFpvF643ypXCWiuddjMKgzuYio6CsAhxZFr5d ~/fobwallet/my
    -keypair.json
Verification for public key: 7frLwuWeFpvF643ypXCWiuddjMKgzuYio6CsAhxZFr5d: Success
```

Now we have create a private key/secret key pair that represents an address on-chain with which we can create accounts that will hold our smart contracts. We can see that nothing is in our wallet yet, meaning no data or account is stored at that address yet:

```
$ ~ spl-token accounts
None
$ ~ solana balance
0 SOL
```

Now we are able to make a non-fungible token. The spl-token program is the Solana Labs SDK that allows for the creation of fungible and non-fungible tokens. Anyone could create new contracts for fungible and non-fungible tokens, but these are the industry standard and have been security tested more so than any other invocation.

In order to make an NFT, the spl-token program must commit a series of transactions. The following diagram shows what a transaction looks like in Solana.



Let's assume only one signature is required (there are cases where multisigs are required for different types of transactions, but that is beyond the scope of this paper). There are two things that are crutial about these transactions. Firstly, account addresses need to be pre-specified. Since the main goal Solana is speed and security, they do away with abstraction such that users must pre-specify all accounts they use in a transaction before actually invoking the transaction. These account addresses can either require a signature to be mutated or not, and are either being read and write to or just read from. Secondly, the instructions represent the actually commands that the transaction is supposed to perform, usually represented in a vector of 8-bit data.

The first step or transaction required in creating a token is through the following command:

```
$ ~ spl-token create-token --decimals 0
Creating token EWSpjToWbVKccty9Uhbz42WhQEuoXn2mK281PMNjacCM

Signature: 42XScFfYHU62kfjnktK9rBahZjcpyBYaqWpwCZDLVnhFKYZtqGzVKxAmqhYHCXhpjv
Hq8pc4eboZTGiTnNecLaV6
```

The source code for this argument is found here.

The first variable is creates is `minimum_balance_for_rent_exemption`, which it checks is whether the minimum balance for rent is exceeded. In Solana, when you create an account to hold a new token, you must also specify the account that has the SOL to pay the "rent" for the account. The rent is the price of account storage on Solana. The owner-controlled state or data of the account is seperate from the account's balance or lamports. Since validators need to maintain a working copy of the states of the accounts on the chain, they charge a rent fee for the consumption of the resource. There's a joke among developers that Solana is a very capitalist blockchain because when an account no longer has it's rent being paid it is immediately "rent collected" and destroyed from the chain.

The next important step is the creation of the instruction data, which is the creation of the "account" and the initialization of the mint authority. The account in this case represents the address of the token we just created, and the initialize mint function creates the authority for the minting. Then these instructions are transformed into a memo type and included in the transaction return data type. Then the code asks for the correct authority to sign for the creation of this token, then the program terminates.

Once the "account" of type token has been created, then comes the instructions to create another account which holds the tokens.

```
$ ~ spl-token create-account EWSpjToWbVKccty9Uhbz42WhQEuoXn2mK281PMNjacCM
Creating account RjYRNDkcABXJ4kZffA2ZjrcfkdcQWyLe34zJZP2xxk9

Signature: 3K9y75WucocsULBYKXe5RMyjgsDrAeWKHkm5QJLk3anuyDMDWEwDPVMD7hPjRw5Weko
9gvmUHQK5ppKmUM3Wz2VN
```

Notice here, when you create the account that holds the tokens you must specify the address of the tokens themselves. This is because in the source code if you specify the associated token address for the account, it will jump to the function `get_associated_token_address` which uses specific instructions to generate the appropriate amount of space the account will need on the chain and initialize it. It is of note that you can also create an account that doesn't hold any tokens, however, you cannot create an account without specifying what type of token/data it should hold, then try to transfer tokens into it. Instead you must first specify all the accounts which you intend to use before you invoke them. This means that if you create an account, you must know before hand the type of data you want it to hold.

The next step is to now mint the token into the account. You specify the address of the token, the amount and then the account where you want the token to reside.

```
~ spl-token mint EWSpjToWbVKccty9Uhbz42WhQEuoXn2mK281PMNjacCM 1
    RjYRNDkcABXJ4kZffA2ZjrcfkdcQWyLe34zJZP2xxk9
Minting 1 tokens
  Token: EWSpjToWbVKccty9Uhbz42WhQEuoXn2mK281PMNjacCM
  Recipient: RjYRNDkcABXJ4kZffA2ZjrcfkdcQWyLe34zJZP2xxk9

Signature: BSUWvrpXf22Mb96eoHN4kzjsZXiDFvKR1N9wCthf2Pa8XTWYkDiab8q2FMDaWXo7ZgLfn
oBbSUqLRTECdQvNy3y
```

The following source code specifies that the command mint calls the function mint-to which returns an instruction to transfer the account of the token, then the command-mint executes the instruction.

Then we must alter the mint authority, which is the owner of the account that is the only one who is able to mint/create new tokens/alter the tokens in any way. Since this is a non-fungible token we have to disable the mint authority such that we change the mint authority from the current wallet address to None.

We created the following NFT through the metaplex documentation.

```
spl-token authorize EWSpjToWbVKccty9Uhbz42WhQEuoXn2mK281PMNjacCM mint --disable
Updating EWSpjToWbVKccty9Uhbz42WhQEuoXn2mK281PMNjacCM
  Current mint authority: 7frLwuWeFpvF643ypXCWiuddjMKgzuYio6CsAhxZFr5d
  New mint authority: disabled

Signature: 4yUtReLc7GLtGngTjUhcsTGk9oo9nNgb5tnfUZ8bU8oCaae1Ma9UvLYut4Uy33iJjEaes
KEzx1a6UjgJ9nDSZNjz
```

Now we see that we have created an account that holds the singular NFT we minted. This account is at the address of the account we made previously and is owned by our own wallet address which we made in the beginning. We also notice that the balance of our wallet has decreased because we needed to pay gas fees for the account.

Now we have created a token that is the only one of it's kind! Now no one will be able to ever create another one like it. However, this is not that useful as it does not contain any metadata or cool information.

Adding metadata, or manipulating these contracts is notoriously difficult in Solana. There is one protocol called Metaplex that is the industry standard for all NFTs on solana. The majority of marketplaces and sites use their contracts as a basis for auctions, and minting. I'm going to be focusing on the token-metadata toolkit found within the metaplex code that represents the contracts for the creation of metadata accounts and then the subsequent tagging of those accounts to the tokens.

The nft-candy-machine of metaplex is a repository of code that allows users to do NFT drops and mint thousands of NFTs simultaneously. When you create a candy machine in metapex, you are creating the capacity to mint NFTs. We are going to focus on how the minting incorporates the metadata fields. We can see here that the metadata is just stored in another account:

```
let metadata_infos = vec![
            ctx.accounts.metadata.to_account_info(),
            ctx.accounts.mint.to_account_info(),
            ctx.accounts.mint_authority.to_account_info(),
            ctx.accounts.payer.to_account_info(),
            ctx.accounts.token_metadata_program.to_account_info(),
            ctx.accounts.token_program.to_account_info(),
            ctx.accounts.system_program.to_account_info(),
            ctx.accounts.rent.to_account_info(),
            candy_machine_creator.to_account_info(),
        ];
```

However, these metadata accounts are an interesting type of account because they are not created specifically by the user, but are actually spun up by other programs, or accounts that are executable. These accounts are owned by the program which creates them, thus the address is not specified by a key-pair. Instead it is specified by a "program derived address" or PDA, where the account struct remains the same, the type of address is different. For example, let's say that you wanted a program to create an account using it's private, public key pair. That would then mean that somewhere in your code you would have to type out and specify what the private key is for that wallet. That is a major security breach and would cause many problems, so the solana-labs developers created a different type of address. The program derived address functions are created by the program specifying a seed phrase and a bump number, such that when the program wants to find an address to store the new account, it does so through hashing the seed, finding a starting

place in the memory and using the bump function to find the first free/available address on-chain. Thus, the address of the metadata accounts are not created by the owner specifically but are created by the executable account, or program. In abstract, this just basically means when you create an NFT on Solana, you must first make the token, then make an account for that token, then if you want to add metadata, that information must live in another account by which the previous account signs for. In the create candy-machine instructure, it created a PDA for the metadata here, using the "invoked_signed" function, which called the function to find a PDA.

The metadata accounts are then created with data of the type :

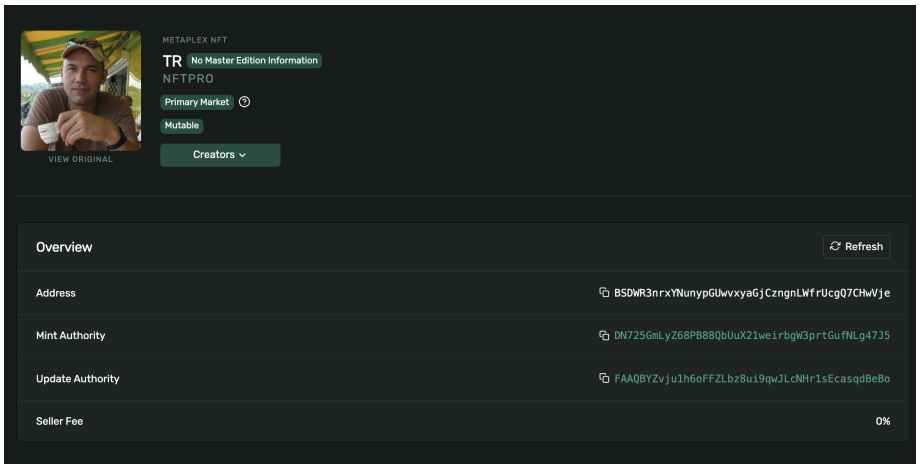
```
pub struct Data {
    /// The name of the asset
    pub name: String,
    /// The symbol for the asset
    pub symbol: String,
    /// URI pointing to JSON representing the asset
    pub uri: String,
    /// Royalty basis points that goes to creators in secondary sales (0-10000)
    pub seller_fee_basis_points: u16,
    /// Array of creators, optional
    pub creators: Option<Vec<Creator>>,
}
```

The uri represents a url which points to a json object that holds any specific metadata you desire about the NFT like a link to the image, or different attributes it might have. As mentioned earlier, since storage is so expensive on Solana, the common practice is to store data off-chain on Arweave or IPFS.

3 NFT Implementations

3.1 Our Implementation

We minted the NFT on solana. We didn't implement one on Ethereum because it would have cost $400 in gas fees! The NFT on the Solana blockchain and can be found here on the Solana block explorer.



You can notice that the metadata is in the same format as the struct Data above:

```
"data":{5 items
```

```
"name":string"TR"
"symbol":string"NFTPro"
"uri":string"https://www.arweave.net/nWOjgg6XmsUBEr1muD02TeqF_SDdClwLm5ulShlrhDQ?ext
    =json"
"sellerFeeBasisPoints":int0
"creators":[1 item
0:{3 items
"address":string"FAAQBYZvju1h6oFFZLbz8ui9qwJLcNHr1sEcasqdBeBo"
"verified":int1
"share":int100
}
]
```

Furthermore, the arweave.net link send us to a json encoding of other data about the NFT:

```
{"name":"TR","symbol":"NFTPro","description":"TR","seller_fee_basis_points":0,"image
    ":"https://www.arweave.net/daOY1vlmsj_U-7xJnMvMnxw2BR3w9AUp8Sr6owwEW_8?ext=jpg
    ","attributes":[],"properties":{"files":[{"uri":"https://www.arweave.net/
    Hm93wp9IF1T4Qftynwef-X_B24_pU7wOuJht4a1mc9g?ext=jpg","type":"image/jpg"}],"
    category":"image","creators":[{"address":"
    FAAQBYZvju1h6oFFZLbz8ui9qwJLcNHr1sEcasqdBeBo","verified":true,"share":100}]}}
```

4 Limitations and Issues

The challenges associated with NFTs can largely be thought of through the lens of the challenges faced by blockchains as a whole. First we point out the issue of scalability. As the number of transactions grows, the gas price grows with them, and if the protocol can only handle a limited number of transactions per second it will end up failing with latency issues. Particularly with NFTs, as the excitement surrounding them and the demand for them grows, we can envision more competition. Given that each token is unique and tied to an individual asset, there will be competition between parties over a specific token, and the more competition there is the harder that can be to process.

In a different vein, NFTs often run into storage issues in which they are forced to be stored off-chain. If a digital art file is too large, it will not be stored on the blockchain. Instead, it will be compressed and housed at a web address, to which the token will point. Not only does this leave the asset vulnerable to link rot or other internet related issues, but it also leads us into the issue of essentially being able to right click and copy an asset image and save the duplication.

There are many fraud and cybersecurity risks surrounding NFTs. Not only can nefarious parties create replica stores which appear similar to original NFT stores in logo and content, they can also create fake NFT stores to sell NFTs that do not exist. Similarly, a party can impersonate a famous NFT artist and commit copyright theft, fake giveaways, NFT replication, etc. This broaches another important topic: intellectual property rights. In order to ensure that a buyer is not purchasing a fake or a copy, they must determine whether or not the seller actually owns the NFT, terms and conditions of which are usually laid out in the metadata of the smart contract. In the future, it is likely that there will be more restrictions put in place such as copyrights and trademarks, ensuring that buyers can only display NFTs of which they are the sole owners.

Other prominent risks on the NFT market are those of smart contracts and NFT maintenance. If smart contract security is inadequate, hackers can attack a DeFi network and steal large amounts of NFTs. We can here take a deep dive into a particularly famous hack from August 2021, the Poly Network exploit. Over $600 million in digital assets was hacked in this NFT theft by exploiting a

flaw in the contract calls, which was done as follows. The network has a manager contract, *EthCross-ChainManager* that is able to trigger messages from other blockchains, and it has the ability to perform cross-chain transactions, which are validated and added to the blockchain. The cross-chain transaction function can also be used to call on another data contract *EthCrossChainData*, which sets and manages a list of public keys that were used to authenticate data (wallets) coming in from other chains. *EthCrossChainData* essentially has the power to determine who (what public key) has the ability to move assets contained in those wallets. The problem is that *EthCrossChainManager* is the owner of *EthCrossChainData*, so it can execute functions within it. Similarly, the input field of part of the *EthCrossChainManager* target contract is user-defined, so if an attacker can brute force the input to *EthCrossChainManager*, they can change the public key and steal the assets in a wallet [9]. This famous example shows how imperative complete security is when it comes to protocols' smart contracts.

Within the NFT market itself there is no standard for determining the value of an NFT, and it will depend on the scarcity of the buyers and owners. This uncertainty makes evaluating an asset a big challenge. Similarly, this kind of distributed ledger technology can present significant challenges around money laundering. Given their decentralized nature, there is no jurisdictional regulation on an NFT, and these decentralized transactions without any monitoring and with support for peer-to-peer transactions allows for money laundering and futures trading issues.

This leads us to look at the legal challenges of NFTs. The objects themselves do not even have a legal definition, so setting regulations and creating laws is very difficult. Different countries classify them differently. For example, in Japan, the classification depends on its capability to serve an economic function, so different assets have different classifications. On the other hand, NFTs do not qualify as legal tender in Singapore at all. This lack of consensus shows the need for an international definition for better regularization and legalization; however, this is an ever increasing challenge as new varieties of NFTs hit the market every day. One such unique variety is NFTs as securities. Legally, NFTs have to comply with the Howey Test to become elligible as securities given that the Supreme Court has already associated them as investment contracts. This means that the NFT and its participants and platforms have to register as a security, securities broker-dealers, and securities exchanges respectively. This action significantly centralizes an NFT investment [10].

The final issue we will discuss here is that of energy usage. This issue is not unique to NFTs, but their purchases and sales are associated with large amounts of blockchain transactions, high energy usage, and greenhouse gas emissions. Particularly in Ethereum, the PoW required to regulate and verify transactions uses a lot of electricity, and the carbon footprint is largely associated with the behavior of blockchain miners and their mining equipment. This issue is actively being improved upon, the details of which we will discuss in the next section.

5 Future Developments

So what is in store for the future of NFTs? As is probably instinctual, a lot of the future developments go hand in hand with those of blockchain technology itself. A lot of these technological developments center around scalability, with the hope of speeding up block time and increasing block size while still dropping transaction fees and keeping the decentralization properties that are fundamental to the blockchain. These ideas are being explored by a multitude of protocols everyday in many ways. Some change the consensus algorithm, some introduce *ZK-SNARKS*, some introduce sharding. The limits of these changes are being tested, but the important thing to remember is that

these parameter changes cannot be so extreme that regular users are unable to run a node. This paper does not go into the details of limitation issues surrounding computing power, bandwidth, and storage of a full node, but Vitalik Buterin (cofounder of Ethereum) has written extensively on it [11].

As we laid out above, a main bottleneck is storage size. Proposed solutions to this problem are statelessness and state expiry. The statelessness property allows nodes to verify the chain without maintaining permanent storage. State expiry, on the other hand, forces users to provide proof to renew states that have not been recently accessed, otherwise it pushes them out. Not only do these solutions alleviate the storage problem, but they also increase the gas limit. Another potential solution to this problem is the implementation of ZK-SNARKS to verify transactions, allow regular users to not personally store the state or verify blocks. In another vein, protocols can implement sharding to decouple the data on the blockchain from the data that an individual node needs to store, so the nodes can verify blocks indirectly; however, this requires more cryptographic complexity. Ethereum is working to incorporate sharding to fix their scalability bottleneck. This change would help to fix the storage issue and associated vulnerabilities of NFTs described above, and would make processing transactions and maintaining assets much easier.

NFTs themselves are becoming major players in other fields of technology, and are poised to grow even farther. One of the most common examples is gaming, in which NFTs are changing how in-game marketplaces operate. Games are being built on blockchain and integrating NFTs into them in many ways, such as rewards for playing. Another interesting place that NFTs are coming in hot is the newly created metaverse. Meta wants to create a digital world in which you can interact with others completely, and introducing NFTs to the space enables the same non-fungibility that we experience with physical world assets, driven by scarcity and utility. Utilizing NFTs within the digital world allows metaverse companies to tokenize all kinds of real world assets, as well as in-game properties such as usernames. With the proof of ownership, transferability, and scarcity inherent to NFTs, game players can view these purchases as not just in-game purchases for regular play, but rather as a real-life investment that provide the potential for future returns.

Another industry being disrupted by the rise of NFTs is ticketing. NFT ticketing goes beyond the ability to attend live events like a standard physical ticket. It now provides the opportunity for holders to get access to lifetime front row seats, or to relive an experience. It can function as a digital asset providing not only a ticket, but an exclusive coexisting investment with an extra incentive for buyers (think artist signature).

Artificial Intelligence-created artwork is not necessarily a new phenomenon; however, with the rise of NFTs, the fame and value of these digital assets is becoming more widely recognized and accessible. Combining the two disruptor fields of AI and blockchain allows AIs to create contacts and mint NFTs from their own digital wallets. The future of this particular field lies in intelligent NFTs. Essentially these are AI personalities that live on the blockchain, with which owners/users can have conversations. Combining these iNFTs with the metaverse described above could allow users to completely live and interact with entities and assets living on the blockchain, creating a full virtual world that has real-world physical value.

The last future application of NFTs we will explore here is related to health care. NFTs are currently being used to allow people to monetize their personal data in the form of a digital asset. Ethereum even has a standard proposal for insurance policies as NFTs [12]. An ongoing wellness NFT project called Health Hero [17] allows users to develop a dynamic NFT that grows as the user exercises, eats well, etc. So, the higher wellness life you live, the rarer and more valuable the NFT

becomes. Blockchain technology is poised to transform the healthcare by orienting the ecosystem around the patient and providing another layer of privacy and interoperability to the data.

6 Culture

Since their rise in 2018, NFTs have had a profound impact on entertainment and artistic culture. They are creating a more flexible and far reaching definition of art, and giving creators another platform on which to promote their work, providing a way for digital art to assert specific creators and owners. It also allows individuals to retain more control and power over the notoriety of their work as well as how it is distributed and sold, potentially bringing a new era of self-funding for artists.

Not only are artists getting the opportunity to sell and promote their work on a new platform, but the architecture of smart contracts also provides the opportunity for creators to participate in the rewards of the platform. Comparing blockchain platforms to other social media platforms on which creators can post content, we can see blatant examples of how NFTs benefit artists. Looking at social medias such as Twitter and TikTok, artists and creators are often not credited for their work and do not share in the royalties of its popularity. Now, NFTs have remapped how creators can interact with and participate in the economy by showcasing what they have created of value, and receiving that value back from other users that interact with it.

NFT issuers such as Nifty Gateway and MakersPlace allow users to purchase assets with standard resources, such as credit cards, though other issuers require users to have crypto digital wallets and transfer other cryptocurrencies in order to purchase and transfer an NFT to their wallet. Although at the moment that extra technical hurdle may be viewed as a headache to the standard consumer, it only further ensures that in the future the percentage of people with stake in the blockchain community will grow, because in order to purchase an asset this way you will need to immerse yourself in the technology.

A problem that we see with the cultural valuation of these NFTs is that there is no standard. As we discussed above, the requirement of cryptocurrency to buy NFTs indicates that their value is fairly dependent on that of the larger crypto market, which we have seen is easily influenced by cultural, political, and economic events (for example, a billionaire tech CEO can tweet about Dogecoin and disrupt the entire market). If we consider the consequences of this association, we can see how harmful it can be. The market is so easily manipulable by world events that a new COVID variant can quickly cause a rapid drop in the value of one NFT while it could raise the value of another. Since there is no way of standardizing the values or stabilizing their fluctuations, it provides an unpredictable and often untrackable income for creators. The relationship between NFTs and cryptocurrency markets may well change and develop over time, especially as traditional market places or established companies (such as eBay or Instagram) begin to appreciate NFT potential. The future may hold some NFT submarkets that are completely independent of the cryptomarket, as we have seen Nifty Gatway and MakersPlace start to inch toward already, further integrating NFTs into the standard interface of everyday life.

7 Conclusion

At the end of the day, NFTs are still an inchoate investment/art form. They have already started to make great strides in creative industries and redefining how we view ownership and collectibles,

and they have quickly become an integral part of popular culture. They are helping to flesh out more realistic digital worlds in which we can interact with others and cultivate our own collections and investments, and they are doing all of this within the first 5 years of mainstream use.

Their explosion onto the scene indicates that as much as has already been done with them, there is even more innovation to come in the future. Our paper has laid out the technical details of how NFTs are created and how they are implemented on two of their most popular platforms: Ethereum and Solana. We also went into the difference between the two, and showed a physical implementation of an NFT on Solana, which we coded ourselves and deployed onto the platform. This shows how easy it is for regular users to create and deploy their own NFTs, thus interacting with the market and greater blockchain community as a whole.

We also went into the limitations of NFTs, and where they run into issues in both a technological and legal sense. This shows that there are still improvements to be made to overall blockchain technology as well as legalization and regulation of NFT valuations. There is a sharp lack of academic material surrounding the development and technological implementation and improvement of NFTs, and it made indepth research more difficult. Reliable sources were difficult to find, particularly due to the heavy saturation of biased news articles drowning out legitimate research. Even disregarding that, we were able to discover and discuss the details of non-fungible tokens and how they relate to both developing standards of technology in art as well as blockchain technology as a whole.

8 Appendix

Code to create an NFT without metadata using JavaScript on Solana. Follows what was done on the command line.

```
var web3 = require('@solana/web3.js');

var splToken = require('@solana/spl-token');

(async () => {
  // Connect to cluster
  var connection = new web3.Connection(
    web3.clusterApiUrl("devnet"),
    'confirmed',
  );

  // Generate a new wallet keypair and airdrop SOL
  var fromWallet = web3.Keypair.generate();
  var fromAirdropSignature = await connection.requestAirdrop(
    fromWallet.publicKey,
    web3.LAMPORTS_PER_SOL,
  );
  //wait for airdrop confirmation
  await connection.confirmTransaction(fromAirdropSignature);

  //create new token mint
  let mint = await splToken.Token.createMint(
    connection,
    fromWallet,
    fromWallet.publicKey,
    null,
    0,
    splToken.TOKEN_PROGRAM_ID,
```

```
  );

  //get the token account of the fromWallet Solana address, if it does not exist,
    create it
  let fromTokenAccount = await mint.getOrCreateAssociatedAccountInfo(
    fromWallet.publicKey,
  );

   // Generate a new wallet to receive newly minted token
   var toWallet = web3.Keypair.generate();

  //get the token account of the toWallet Solana address, if it does not exist,
    create it
  var toTokenAccount = await mint.getOrCreateAssociatedAccountInfo(
    toWallet.publicKey,
  );

  //minting 1 new token to the "fromTokenAccount" account we just returned/created
  await mint.mintTo(
    fromTokenAccount.address, //who it goes to
    fromWallet.publicKey, // minting authority
    [], // multisig
    1, // how many
  );

  await mint.setAuthority(
    mint.publicKey,
    null,
    "MintTokens",
    fromWallet.publicKey,
    []
  )

  // Add token transfer instructions to transaction
  var transaction = new web3.Transaction().add(
    splToken.Token.createTransferInstruction(
      splToken.TOKEN_PROGRAM_ID,
      fromTokenAccount.address,
      toTokenAccount.address,
      fromWallet.publicKey,
      [],
      1,
    ),
  );

  // Sign transaction, broadcast, and confirm
  var signature = await web3.sendAndConfirmTransaction(
    connection,
    transaction,
    [fromWallet],
    {commitment: 'confirmed'},
  );
  console.log('SIGNATURE', signature);
})();
```

References

[1] Boscovic Research Professor of Computing, Dragan. "How Nonfungible Tokens Work and Where They Get Their Value – a Cryptocurrency Expert Explains Nfts." The Conversation, 29

July 2021, https://theconversation.com/how-nonfungible-tokens-work-and-where-they-get-their-value-a-cryptocurrency-expert-explains-nfts-157489.

[2] https://www.trustnodes.com/2021/10/06/nfts-near-5-billion-in-market-cap

[3] Adesina, Olumide. "Ethereum losing Steam To Solana, Cardano, Polygon", 3 October 2021. https://www.yahoo.com/now/ethereum-losing-steam-solana-cardano-044353270

[4] Lipton, Alex, and Stuart Levi. "An Introduction to Smart Contracts and Their Potential and Inherent Limitations." The Harvard Law School Forum on Corporate Governance, 26 May 2018, https://corpgov.law.harvard.edu/2018/05/26/an-introduction-to-smart-contracts-and-their-potential-and-inherent-limitations/.

[5] https://github.com/minimalsm, Joshua. "How to Mint an NFT (Part 2/3 of NFT Tutorial Series)." Ethereum.org, 25 Nov. 2021, https://ethereum.org/en/developers/tutorials/how-to-mint-an-nft/.

[6] https://github.com/minimalsm, Joshua. "ERC-721 Non-Fungible Token Standard." Ethereum.org, 2 Dec. 2021, https://ethereum.org/en/developers/docs/standards/tokens/erc-721/.

[7] William Entriken, Dieter Shirley, Jacob Evans, Nastassia Sachs, "EIP-721: Non-Fungible Token Standard," Ethereum Improvement Proposals, no. 721, January 2018. [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-721.

[8] "Cryptokitties Craze Slows down Transactions on Ethereum." BBC News, BBC, 5 Dec. 2017, https://www.bbc.com/news/technology-42237162.

[9] Gagliardoni, Tommaso, and John says: "The Poly Network Hack Explained." Kudelski Security Research, 12 Aug. 2021, https://research.kudelskisecurity.com/2021/08/12/the-poly-network-hack-explained/.

[10] Lopez, Maria. "Mention Risks and Challenges Associated with Non Fungible Tokens (Nfts)." Medium, DataDrivenInvestor, 18 Sept. 2021, https://medium.datadriveninvestor.com/mention-risks-and-challenges-associated-with-non-fungible-tokens-nfts-c18b933a008f.

[11] Buterin, Vitalik. "The Limits to Blockchain Scalability." Vitalik Buterin's Website, 23 May 2021, https://vitalik.ca/general/2021/05/23/scaling.html.

[12] Christoph Mussenbrock, "EIP-1523: Standard for Insurance Policies as ERC-721 Non Fun-gible Tokens," Ethereum Improvement Proposals, no. 1523, October 2018. [Online serial].Available: https://eips.ethereum.org/EIPS/eip-1523.

[13] https://spl.solana.com/token

[14] https://www.quicknode.com/guides/web3-sdks/how-to-mint-an-nft-on-solana

[15] https://twitter.com/pencilflip/status/1463284754841681920?s=21

[16] https://docs.metaplex.com/nft-standard

[17] https://enjin.io/showcase/health-hero