Title: Let's talk crypt0 dividends; I'll explain Bchain smart contracts, argue why we might not get NFT dividends, but fungible crypt0 tokens instead, and finally, you will help me fill in some gaps
Author: Best_Account
Created 2021-10-13 17:23:38 UTC
Permalink: /r/DDintoGME/comments/q7g1m1/lets_talk_crypt0_dividends_ill_explain_bchain/
Url:
https://www.reddit.com/r/DDintoGME/comments/q7g1m1/lets_talk_crypt0_dividends_ill_explain_bchain/

TL;DR: Creating a single NFT cost is high. Might not be cost-effective to create **76.49M NFTs**. We might be wrong about NFT dividends. We'll do some simple math further down.

Edit: Thank you for all the upvotes. I'll repost this tomorrow when market opens on bigger subs. Please help me to get a word out by upvoting those post tomorrow. It takes a bit more effort to gain traction on those subs

If you're not familiar with Bchain and smart contracts, I wrote an ape-friendly post on the subject (it's on the bigger GME sub and I can't link it here you can find in my post history under the title "MOASS is a valid reason to hodl, but here's a reason to hodl beyond...") some months ago. I've been playing around with smart contracts for about a year in my spare time and I'd like to share some details which might help us clarify certain things.

I'm still learning about smart contracts and I might be wrong or missing something. I'd like to hear other apes with wrinkles on this subject. I'd like to also hear more details on how it went down with Overstock if there's more to it. Lastly, this came to my mind yesterday, and I thought I'd share ASAP this in case GameStop comes out with an NFT marketplace these coming days, but we don't get dividends. Do not FUD!

I'm a button-pushing ape, not a financial advisor. If you just read the linked post above, I apologize if I repeat myself. Lastly, English isn't my first. Please bear with me.

We will look into different token standards a bit more in this post; fungible tokens (*ERC20*), non-fungible tokens (*ERC721*), and a combination of both (*ERC1155*). But before we continue I'd like to note that it costs money to store data on Bchain. Let's say you're a smart contract developer, you wrote a smart contract and decided to deploy it on the Bchain. There is a smart contract deployment fee. Smart contracts, just like your regular apps and websites such as Reddit have functionalities; whenever you write a post and click "Post" button there's a function in the background that sends all the data to the database. Whenever someone visits the page with your post, the data is requested from the database and shown to the ape that visited your post. Same with Bchain - we have functions running certain commands inside a Bchain. Two types of functions; those that request data from Bchain and those that store data on the Bchain. Requesting data from Bchain has no cost, but there's a fee for storing, and obviously the more data you store, the higher the fees that you have to pay. So there's also a fee for every function execution that stores data on the Bchain.

Now let's go back to the tokens. We will mostly focus on the minting (creating tokens) process:

When smart contract developers write the code, they can create a function that gives the ability to mint (create) tokens inside of a smart contract. As we know we have different types of tokens standards therefore each token standard has unique qualities and is minted with different conditions.

ERC20

Fungible tokens - this is the most simple one to mint. Here's a very simple example of a minting function [from OpenZeppelin](https://docs.openzeppelin.com/contracts/2.x/erc20-supply):

```
contract SampleApeContract is ERC20 {
constructor() public {
_mint(msg.sender, 100000);
}
}
```

Here we have a SampleApeContract which inherits functions from ERC20 (give's the ability to mint). Next, inside of a constructor (function) we _mint 100000 tokens and the Bchain sends it to the address of whoever deployed the contract (msg.sender). We're only providing two pieces of data to the minting function - who should receive the minted tokens and the number of tokens to mint. With some minor changes to the code above we can create similar tokens to 50.000.000 [WorthlessCramer tokens,](https://etherscan.io/token/0x71d4ccd6a41d45f785000500231f4bbe06df2a01) I created in the previous post. The cost does not really change if you mint 1000 tokens or 10000000, and you can mint them all at once.

# ERC721

Non-fungible tokens - the [example from OpenZeppelin](https://docs.openzeppelin.com/contracts/3.x/erc721) is longer to make a full breakdown in this post and it's a little more complicated. Instead, I'll just explain a couple of lines of code from that example that are responsible for minting ERC721 NFTs:

```
_mint(player, newItemId);
_setTokenURI(newItemId, tokenURI);
```

When we _mint ERC721 (NFTs) we're providing two pieces of data - player (identified by an address), and newItemId which incrementally generated unique ID (somewhat similar to ComputerShare account numbers).

_setTokenUri is responsible for storing a link to the NFT. Now, this might be something new and seem complicated for some apes, but those images or whatever you see presented to you as NFT, the actual file representing the item you're looking at is not stored on Bchain. As an example, storing so much data on Bchain that consists in an image file would cost a lot in fees. Instead of storing good old image or video files as we're used to storing them on our hard drives, NFTs are stored as links to metadata on the Bchain. Here's an example of metadata from the OpenZeppelin NFT721 page linked above:

```
{
"name": "Thor's hammer",
"description": "Mjölnir, the legendary hammer of the Norse god of thunder.",
"image": "https://game.example/item-id-8u5h2m.png",
"strength": 20
}
```

It's straightforward; we have the NFT item name, description, and **a link to an image** inside of the metadata. Files are normally stored in decentralized storage solutions such as IPFS instead of Bchain, thus NFT metadata files and all related media files are stored off the smart contract and Bchain.

Let's recap: we mint an NFT token using those two lines of code above. _mint function accepts two arguments, player = address the created token should be sent to, and newItemId = unique ID. Next line we store the link to the metadata with all the information related to that particular NFT token.

There's one big difference between _mint function on ERC20 and on ERC721. With ERC20 we indicated how many tokens we would like to create (100000), but ERC721 _mint function expects no such input because each NFT token is meant to be unique. As you can guess, we can have different data recorded in each metadata which makes each token different from any other token. So what we have here is a limitation to create a single NFT at a time and each time we execute the ERC721 _mint function we're storing data on Bchain thus we pay a network fee. At the time I created an NFT it cost me around $50, but fees went up since AFAIK ($60-$80). Fees vary based on network factors. Eth devs are working on solutions to bring the fees down, but even if fees are down by ten-time and network fee goes down to $6-$8, don't you think it'd be a bit expensive to mint **76.49M NFTS for every GME share?** We're talking almost **$500M in network fees** to mint **76.49M at $6 per NFT.**

Two things worth mentioning before we move on; lazy minting, and hacking batch minting function.

One way to avoid fees upon minting is a method called "lazy minting." Let's take an artist with a digital artwork image as an example. With regular minting, both minting on Bchain and off-chain metadata with the image file uploading process happen around the same time. With lazy minting, the image and the metadata is uploaded off-chain but the NFT is not minted on the Bchain. Instead, the token is minted whenever someone purchases the token and the network fee is paid by the purchaser not by the artist. I believe this method is worth mentioning, but I also believe it won't work well with NFT dividends since beneficiary shareholders would have to pay the minting fee themselves in order to receive the NFTs.

Another thing worth mentioning; if I remember correctly, there is a way to hack a function together which allows you to mint more than one NFT per transaction with ERC721 token standard. While that might help reduce the number of times you're manually executing the mint function, the minting fees are not eliminated and added up together as a single big fee.

Last note before we move on; if you look at ERC20 tokens process and think about it, you will realize that there is no additional data stored on the Bchain with each created token. Each token is fungible and essentially represents an identical value. Whereas ERC721 tokens are non-fungible and each token is unique. Each token has additional data included when minted. This makes it more expensive to mint NFTs. You're storing more data with each created NFT on the Bchain.

# Overstock

u/derrida_n_shit pointed out to me in another post that Overstock issued **OSTKO** crypt0 tokens as a dividend. I don't know much about Overstock dividends other than what most other apes know - Crypt0 dividends were issued to counter the shorting. I googled Overstock NFT dividends but couldn't find anything significant and as far as I can tell, according to [this page](https://stomarket.com/sto/overstock-ostko) OSTKO token is a fungible ERC20 token. Correct me if I'm wrong, but Overstock never issued NFTs. They did issue "Crypt0currency" ERC20 tokens, but that's not NFT.

Please share more on this if you have any helpful information.

According to [this](https://marketrealist.com/p/overstock-digital-token/):

>For every 10 shares you owned, you would receive one OSTKO token as a dividend.

If we calculate the minting cost of 7.6M NFTS for GME we end up with close to $50M in fees - that's if we go 10 shares per token route and don't forget, that we lowered the true fee by 10 times. If nothing changed in the past several months I haven't been keeping an eye on the network updates, the fees should still be around $60-$80 per mint. So even if we go 10 to 1, with true minting fees it would cost hundreds of millions of dollars to mint 7.6M NFTs.

Based on the current conditions of the Eth network and on everything above I believe we might be wrong about NFT dividends at the moment. Not only it doesn't make sense when you do the math, but we also have another example of crypt0 dividends that were given out as fungible ERC20 tokens. Before you get upset or call me a shill, don't forget that ERC20 is a Bchain token and it worked for Overstock. To me, it makes more sense to mint ERC20 tokens which cost laughably cheaper to mint, and distribute them as dividends if the result is the same. Hold on! How about the GameStop NFT website? Doesn't really fit in all this... Actually, there are no restrictions to how many contracts someone can create. They could create fungible ERC20 tokens and distribute them as dividends, and they could also create ERC721 for an NFT marketplace that they are working on. Problem solved! But if I was GameStop and I was trying to create a cutting edge technology I'd probably try a different token standard and create something with...

# ERC1155

This token standard provides the ability to mint both fungible and non-fungible tokens in a contract.

Two examples I can provide to help you better understand how it works:

1. Imagine a game where the in-game currency is minted using the fungible tokens, and items in the game are minted using non-fungible. Each minted non-fungible item is unique. A couple of the differentiating characteristics are that they are minted at a different time and used by different players. You can purchase in-game items with ETH coin, but maybe there's some sort of unique utility to the fungible token which gives you a discount or monetizable coin as a reward. For example, some crypt0 exchanges provide lower trading fees if you use their tokens to trade. Another example is DeFi projects reward users for staking tokens.
2. Imagine you have a favorite sports team and they create an ERC1155 smart contract. They create fungible currency tokens and NFTs for items such as player collector cards. You can purchase the cards for any currency you prefer, but get a discount or point for purchasing with their currency.

That's how ERC1155 works - you can create both types of coins in a contract.

There are differences between ERC721 NFTs and ERC1155 NFTs. A couple of differences related to our subject: minting fee which is about 80-90% cheaper with ERC1155, as far as I'm aware. Another difference is that the batch NFT minting function comes with ERC1155 and multiple NFTs can be minted with a single execution.

Oh, and by the way! I thought apes might find this interesting - people from [Enjin](https://enjin.io/) are the creators of the ERC1155 token standard. Here's an excerpt from their website:

>ERC-1155 is the only token standard that enables you to create every type of asset, from currency and real estate to digital art and **gaming items.**

# GameStop Marketplace

This part is speculative and this is purely my thoughts and you should take it with a grain of salt.

Game item NFT is a popular use case that is discussed on Bchain communities and implemented in some games already. I believe this is the next big thing waiting to happen in the gaming industry and GameStop might be working on a smart contract where game developers can mint items for games they're developing. Items can be sold on GS marketplace and you get credit for using GS fungible currency. Btw, when it comes to fungible tokens, 76M of them are distributed to shareholders and they can use them in the marketplace. Let's take it a step further (*the most speculative part*) and imagine that they provide dividends FOR ALL SHARES, even the ones over 76M. Minting fungible tokens don't cost much, remember? So why not reward more apes which most likely are going to come back to their favorite company and spend the dividend tokens on NFT items on the marketplace. This is how everything can be accomplished - Short countering dividends are minted in the form of fungible tokens, the next-gen gaming marketplace is created with loyal apes with tendies and fungible token dividends, and GameStop establishes itself as a dominating player in the gaming tech industry. The part about providing shares to hodlers with shares that are over 76M might be restricted in some way when it comes to regulatory forces outside of Bchain.

# Counterarguments

If we look at [the smart contract](https://etherscan.io/address/0x13374200c29C757FDCc72F15Da98fb94f286d71e#code) from nft.gamestop.com on the eth network, it clearly inherits ERC721 token standards (scroll through the code in "Contract" tab). I didn't do much digging into this but assuming this is the smart contract that will be used as the platform they're working on, my speculations in regards to ERC1155 smart contract are incorrect. Someone can correct me if I'm wrong, but as far as I'm aware you can't change the contract once it has been deployed. I know there's a break down of the contract on subs somewhere, but I ran through the contract really quick and here's my take - you can look through the contract as you read the explanation.

930 lines of code are just inherited contracts, functions, etc of ERC721 token standard. Think of them as the code that sets up ERC721 functionality within the contract. Next, we have the contract itself:

```
contract Gamestop is ERC721 {

address public owner = 0x10B16eEDe03cF73CbF44e4BFFFa3e6BFf36F1Fad;
string public ipfsLocation = "ipfs://QmaLEchFaE7FWhc4MCvYMqoTdK8rV1yfjEC5Bz4jzQRbjS";
uint public launchDate = 1626261600;

event Mint(address indexed owner, uint indexed _punkId);

constructor() payable ERC721("Gamestop", "GME") {
_mint(owner, 0); // The one and only
emit Mint(owner, 0);
}

/**
* @dev Returns a URI for a given token ID's metadata
*/
function tokenURI(uint256 _tokenId) public view override returns (string memory) {
require(_tokenId == 0);
return ipfsLocation;
}

modifier onlyOwner() {
require(owner == msg.sender);
_;
}

function setIPFSLocation(string memory _ipfsLocation) public onlyOwner {
ipfsLocation = _ipfsLocation;
}

function setLaunchDate(uint _launchDate) public onlyOwner {
launchDate = _launchDate;
}

}
```

There's nothing complicated in this contract that indicates that it will be responsible for complex operations. There are only 3 functions, not counting the constructor function. If you look through the names of functions:

tokenUri - Requests ipfsLocation link.

setIPFSLocation - Changes the IPFS link.

setLaunchDate - Changes the launch date

Nothing indicates that this is the contract that will be whatever they are creating.

# Layer 2 and Eth Upgrade

Transaction fees are a big factor in Bchain space among competing Bchains. Eth development team has been working on a solution to bring the fees down. As far as I'm aware there are other developers that came up with solutions to reduce fees when you use the main ETH network in combination with those solutions. Eth dev team itself should come up with an upgrade this November - early December which should reduce fees by 40-90% according to different sources. To be honest I haven't been following the latest news and updates but everything I wrote above was true several months ago. Even if transaction fees are reduced with new updates, it just doesn't make sense to mint NFT dividends with a higher fee, when a fungible dividend leads to the same result when it comes to countering shorts(according to Overstock precedent). But if I am wrong and we will get NFT dividends, it'll be ERC1155 NFTs which are cheaper to mint, and after the scheduled Eth network upgrade in a little over a month. I assume fees are going to drop further down for ERC1155 NFTs.

Anyhow, nothing changes... DRS your shares and HODL.