# 659.https://stackoverflow.com/questions/67751557/unnoticeable-changes-leads-to-different-tokens

**T:**unnoticeable changes leads to different tokens

**Q:**My question relates to the minting process to create an NTF.  
  
I might be wrong but the tokenization function can be compared to an hashing function whichtakes as input the mediaandoutputs the token.  
  
Yeah this actually already is a question, cause otherwise the main question maybe does not makes sense.  
  
Assuming the comparison to an hash function makes sense and forgetting about collisions let's assume the following scenario:I create a digital artwork and the related NFT. It's published and sells somehow (hopefully :D).  
  
Imagine Mr.XYZW is a well known digital artist who gets huge revenues from NFT, he sees my artwork, somehow he likes it butalso thinks the artwork would look better if for example the colors simply get inverted. Here I'm just mentioning one of allthe possible changes he could do, the point is that easily those changes could not even be noticeable to the human eye, but not to the tokenizer,which would in the end clearly create a different token.  
  
Now the problem should be clear.  
  
If what I said makes sense, how is it usually tackled?in case it doesn’t, please help me to understand.  
  
Thank you

**C1:**I’m voting to close this question because it's about legal topics rather than programming.

**C2:**I think You are wrong,... there’s clearly a legal aspect. But this is not what I aimed to clarify

**C3:**I updated the question so to look less legal oriented. thx Petr :)

**C4:**Thanks for the clarification. I retracted my close vote.

1 **Answer**

**A1:**tokenization function can be compared to an hashing function which takes as input the media and outputs the token  
  
This is an incorrect assumption.  
  
You can compare an NFT collection (at least per the most widely used standard - ERC-721) to a key-value dictionary, where the key is an integer ID, and the value is a URL. The standard defines that the URL should lead to a JSON containing the token name, description, and image URL.  
  
But there's no hashing function that would calculate the token parameters based on the image.  
  
Each collection (holding several NFTs) is a smart contract deployed on a different address (e.g. 0x12345). Also, each NFT within its collection has a unique ID (e.g. 1).  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Combination of the collection address and the token ID can be used as a unique identificator of each NFT (e.g. 0x12345 / 1).  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
It's technically possible for multiple different NFTs (no matter whether they're in the same or different collections) to lead to very similar images or even the same image. But the combination of collection address and token ID is always unique.

**C1:**ok thx for the clarification,...this means that the media itself doesn’t matter at all... matters where it’s stored (that must at that point be immutable), token name, description, etc but not the media content.

**C2:**Exactly. Unless the image is stored on an IPFS or a similar storage that is immutable (even though it's somewhat expensive to store images on IPFS), the contract address and token ID are the only immutable parameters. (Note: It depends on the contract creator - it's technically possible to make the ID mutable as well - it would just not follow the standard).

**C3:**also ....and this was the point I was aiming to (regardless hashing.. which does not apply on the media in this case): we could both mint two nft from the same exact media and there will be no way to know who is the author because here ‘author’ is not about the media but about the information the token is pointing to

**C4:**how this is not a huge massive problem?

**C5:**NFTs just prove ownership of the token (representing the art piece) - they are not meant to prove authorhip/ownership of the actual art piece... If you used them to prove some rights to the actual art piece, that would cause the problems that you're referring to in the comment.