# 209.https://stackoverflow.com/questions/72351730/what-is-the-best-way-to-randomize-a-selection-of-minted-nfts-in-solidity

**T:**What is the best way to randomize a selection of minted NFTs in Solidity?

**Q:**I want to randomize how different levels of NFTs are minted. If there are 3333 in total and 20 Tier 1 (being the best), 313 Tier 2, 1200 Tier 3 and 1800 Tier 4, does the following code do the trick? Could this result in the Tier 1 NFTS being minted every other 4 mints?  
  
function \_genTier(uint256 seed) internal view returns (uint256) { uint256 index = seed % probabilities.length; uint256 rarity = seed.mul(100).div(MAX\_SUPPLY); if(rarity < probabilities[index]) { rarity = index; } else { rarity = aliases[index]; } return rarity;}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

1 **Answer**

**A1:**TL;DR if you are already using an oracle this approach is going to work.  
  
The method from the code snippet in your post is going to result in the rarity, but the NFTs would not be assigned at random but in a sequence depending on the seed provided. I am not sure where the seed is obtained from.  
  
Normally, I like to use the "random boxes" technique, where you can create traits boxes, shuffle and loop through them. So you pretty much generate the traits and end up with arrays of, in this case, 3333 items. If you would like to maintain exact rarity, like  
  
3333 in total and 20 Tier 1, 313 Tier 2, 1200 Tier 3 and 1800 Tier 4  
  
it is a good way to perform the generation. Conversely, the method \_genTier might not lead to exactly the numbers you provided, just the likelihood of minting each of the NFTs being as outlined in the rarity.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
However, the "random boxes" method is not optimal for Solidity as the amount of memory required for storing this much data would yield redundant costs which can be avoided by using randomness.  
  
Randomness, on the other hand, is also difficult to obtain on-chain. To get the cryptographically secure random numbers you could use the block hash or an oracle.  
  
Using the block hash might work for this application, but at its core it is predictable. Meaning, that someone could get the random number knowing the upcoming block hash and figure out the tier of the NFT they would be to receive.  
  
Thus, I would advise looking into oracles like ChainLink to get truly cryptographically random numbers that cannot be predicted. The RNG from oracles comes from off-chain and conforms to the security standards. You could use the numbers and include an if-statement that would check the number of currently generated NFTs from each tier. Then, re-roll if say the 20 from the Tier 1 have already been allocated in order to ensure that the final 3333 is exactly as planned.