**Claiming an NFT**

In this assignment, we’ve deployed an NFT contract on the [Avalanche Fuji Testnet](https://support.avax.network/en/articles/6224787-how-to-connect-to-the-fuji-testnet).

The Avalanche Fuji Testnet is an EVM-compatible blockchain, so contract code works in exactly the same way as on Ethereum. The Avalanche Testnet has faster block times, and it’s easier to get funds from the faucet, so it’s more pleasant to test on it.

The NFT Contract is deployed at  
[0x85ac2e065d4526FBeE6a2253389669a12318A412](https://testnet.snowtrace.io/address/0x85ac2e065d4526FBeE6a2253389669a12318A412)

For convenience, we have also provided the

ABI for the NFT contract

. The ABI will be useful if you want to interact with the contract using Python’s web3 library.

The NFT contract we have deployed uses ERC721 NFT tokens. ERC721 tokens have “non-native” requests and responses so, you will have to use “Middleware” to successfully interact with our contract. You can read about Middleware [here](https://web3py.readthedocs.io/en/stable/middleware.html). Once you understand what Middleware does, you will add the “Proof of Authority” Middleware to the Web3 object that you create to interact with the contract. Details about this Middleware are on the same page linked above.

**Assignment**

Your assignment is to mint yourself an NFT with the smallest possible token ID

In order to interact with the blockchain, you’ll need AVAX to pay gas fees. You can get testnet AVAX from a faucet.  
Unfortunately, the [main Avalanche Faucet](https://core.app/tools/testnet-faucet/?subnet=c&token=c) now requires you to have a coupon code,  
or a balance on mainnet to get testnet tokens.

[Quicknode has a faucet](https://faucet.quicknode.com/avalanche), but they’ll only give to addresses that have a balance on the Ethereum mainnet.

[Chainlink has a faucet](https://faucets.chain.link/fuji), and they’ll give you testnet AVAX if you identify yourself with Github.

The contract provides three ways to mint an NFT

1. You can call the “claim” function, and provide a random nonce. The claim function will mint you an NFT with tokenId = keccak256(nonce) % maxId. If that tokenId has already been claimed by another user, then the function will revert.
2. You can call the “combine” function, and provide two tokenIds. combine(tokenIdA,tokenIdB) will mint you an NFT with tokenId = gcd(tokenIdA,tokenIdB). If that tokenId has already been claimed by another user, then the function will revert.
3. You can call the “repossess” function, and provide two tokenIds. repossess(yourTokenId,otherTokenId) will transfer the token with otherTokenId to you if yourTokenId is a divisor of otherTokenId. If otherTokenId has not been minted yet, the function will revert.

In order to pass this assignment, you must prove that you have claimed at least one NFT. In order to prove that you do indeed own the NFT, our autograder will send you a random challenge (a random byte string), and you must sign the challenge with the private key corresponding to the address that owns the NFT.

Concretely, you will need to

1. Get an NFT from the contract
2. Complete the “signChallenge” function in

verify.py

 by adding the secret key corresponding to the address that you used to claim the NFT. You do not need to make any additional changes to the file beyond adding your secret key.

You can claim the NFT in any way you’d like, e.g. with Python, through a wallet, or even through a block explorer. We will only check that your address has the NFT.