# 40.https://stackoverflow.com/questions/73365543/how-to-prevent-users-from-calling-my-smart-contract-function-while-still-being-a

**T:**How to prevent Users from calling my Smart Contract Function While still being able to call it in my dApp

**Q:**I'm currently working on a Web3 Game and I need a function to give the player some coins. However I know that Smart Contracts are easily accessible. From what I understand I can not use onlyOwner as I want to call it from my Dapp, where other users would be connected that are not the contract owner.  
  
// Add Coins to an Accountfunction addCoinsToAccount(uint256 tokenId, uint256 coins) public{ // Add Coins to Account attributes[tokenId].coins += coins;}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
I was going to use a SecretPassword Method However after reading this I now see even this is easily hackable.https://medium.com/coinmonks/a-quick-guide-to-hack-private-variables-in-solidity-b45d5acb89c0  
  
If anyone knows how I could accomplish this it would mean a lot! Thank you for taking the time to help me.

**C1:**Why can't you use onlyOwner like you mentioned and call it from your frontend (after signing with your wallet)?

**C2:**Because how would my dApp run the function when someone else is on the site? My wallet wouldn't be the signed wallet.

2 **Answer**

**A1:**u can sign with a private key in the backend so only user who are "whitelisted" can redeem the coin.

**C1:**Could you elaborate a bit more? Thank you!

**C2:**u can sign a message with ur private key and then check if the signature was sign by u. if the signature is sign by u u can execute the function. u can see how marketplace like opensea do it, when u want to list an nft to opensea u sign a message contening the info like the address of the token u want to sell, the price ect. the smartcontract will check if the signature corresponding to ur address and then transfer the nft. solidity-by-example.org/signature

**C3:**so in ur case u sign a message like how many token the user can redeem with the address of the user and a nonce so the user can call the function just once

**A2:**You could have a Request struct:  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
struct Request{ string description; uint value; address payable recipient; bool complete; }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
so when you need to send coins, you create an object set the complete false. You keep those requests in an array.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Request[] public pendingRequests;// maybe add a modifier onlyPlayers can call thisfunction createRequest(string memory description, uint value, address payable recipient) public { // add require logic to decide who can call it // maybe players who are stored in a mapping, winnign certain rewards can call this. depends on your game Request memory newRequest=Request({ description:description, value:value, recipient:recipient, complete:false, }); pendingRequests.push(newRequest); }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
then you create a finalizeRequests function that can be called by the admin only. Not necessarily the owner. you can add a function to decide who can be the admin. Then admin will go through the pending requests and mark complete property as true.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]