# 218.https://stackoverflow.com/questions/72285587/how-to-verification-address-in-solidity

**T:**how to verification address in solidity

**Q:**Through web3 transactions, the private key is used to sign the initiated transaction to verify the transaction information and prove the owner of the address.  
  
But in the transfer method in the smart contract erc721, I did not see the code related to the signature  
  
Does this mean that as long as I have the address I can make the transaction directly without the need for the private key to sign

1 **Answer**

**A1:**Each smart contract method is executed after it's been verified (by miners/validators) that the transaction has been signed with a private key corresponding to its sender address.  
  
In other words, you don't need to perform any private key validation in your smart contract - it's been done on a lower layer.  
  
Does this mean that as long as I have the address I can make the transaction directly without the need for the private key to sign  
  
You always need the private key to sign a transaction. Emulators used for development (such as Ganache, Hardhat, JS VM in Remix IDE, ...) usually hold the private keys and sign the transactions for you so the signing process might be sometimes hidden, but a valid transaction always needs to be signed with a correct private key.

**C1:**you mean my contract is ownable. The user sends a request to my service (such as java), after my service has verified the signature, the service calls the smart contract to transfer the nft. Is this possible? Is this a general solution? Thank you so much for taking your time to teach me

**C2:**@dazan The signature is validated by the block miner/validator, there's generally no need to perform another validation by an off-chain service. Even though it is technically possible... A common practice is to prepare a transaction for the user on a frontend, have them sign it and broadcast it from their browser (e.g. through MetaMask or WalletConnect), and validate the input params on your smart contract. If the contract needs to communicate with your backend service, it can emit events, and you can handle these events from the backend service.