# 563.https://stackoverflow.com/questions/70005355/mekaverse-nft-smart-contract-is-using-ecdsa-but-i-dont-understand-how-it-works

**T:**MekaVerse NFT smart contract is using ECDSA, but I don't understand how it works

**Q:**In the smart contract of MekaVerse I can see these lines to enable a whitelisting, but I don't understand the theory behind it and how I can use it.  
  
function mint(uint256[] memory \_tokensId, uint256 \_timestamp, bytes memory \_signature) public payable saleIsOpen { uint256 total = totalToken(); require(\_tokensId.length <= 2, "Max limit"); require(total + \_tokensId.length <= MAX\_ELEMENTS, "Max limit"); require(msg.value >= price(\_tokensId.length), "Value below price"); address wallet = \_msgSender(); address signerOwner = signatureWallet(wallet,\_tokensId,\_timestamp,\_signature); require(signerOwner == owner(), "Not authorized to mint"); require(block.timestamp >= \_timestamp - 30, "Out of time"); for(uint8 i = 0; i < \_tokensId.length; i++){ require(rawOwnerOf(\_tokensId[i]) == address(0) && \_tokensId[i] > 0 && \_tokensId[i] <= MAX\_ELEMENTS, "Token already minted"); \_mintAnElement(wallet, \_tokensId[i]); }}function signatureWallet(address wallet, uint256[] memory \_tokensId, uint256 \_timestamp, bytes memory \_signature) public view returns (address){ return ECDSA.recover(keccak256(abi.encode(wallet, \_tokensId, \_timestamp)), \_signature);}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
The interesting part that I don't understand is here :  
  
address signerOwner = signatureWallet(wallet,\_tokensId,\_timestamp,\_signature);require(signerOwner == owner(), "Not authorized to mint")  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
And here :  
  
function signatureWallet(address wallet, uint256[] memory \_tokensId, uint256 \_timestamp, bytes memory \_signature) public view returns (address){return ECDSA.recover(keccak256(abi.encode(wallet, \_tokensId, \_timestamp)), \_signature);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
}  
  
Thank you for your help,Ben

1 **Answer**

**A1:**The MekaVerse contract uses the OpenZeppelin ECDSA implementation, specifically its recover() function. ECDSA stands for "Elliptic Curve Digital Signature Algorithm" and basically, it allows to sign a message using a private key and to check validity of the signature without providing the private key.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
The recover() function takes 2 arguments in this case: bytes32 (array of 32 bytes) hash of a signed message, and bytes (dynamic-length array of bytes) signature. Then it validates whether the hash and signature match according to the ECDSA. If it does, it returns the signer address. If the validation fails, it returns the zero address (0x0).  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Note that the signature is a result of signing a message using a private key - but it's not the private key.  
  
You can learn more about signing messages in the web3 documentation of the sign() function. If you're interested in the ECDSA (or cryptography in general) in more depth, the wiki page shows some basic information and links to other sources.