# 299.https://stackoverflow.com/questions/71732049/using-hashtypeddatav4-in-smart-contract-and-signtypeddataetherjs-in-frontend

**T:**using \_hashTypedDataV4 in smart contract and \_signTypedData(etherjs) in frontend is not working

**Q:**i am trying to encode the nft data using \_signTypedData(etherjs) in frontend as follows  
  
 const domain = { name: "og-nft", version: "1", }; const types = { Nft: [ { name: "URI", type: "string" }, { name: "price", type: "uint256" }, ], }; // The data to sign const [voucher, setVoucher] = useState({ URI: "", price: '1', });const signature = await signer.\_signTypedData(domain, types, voucher);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
reference to above\_signTypedData in docs  
  
I am storing the voucher and signature in the mongo database, I have deployed smart contract on hardhat and I am verifying the authenticity of signature by peering out the signer of the voucher using ECDSA.recover  
  
function verifyVoucher(NFTVoucher calldata voucher, bytes memory signature) public view returns (address) { require(voucher.price > 0, "Price must be greater than 0"); // require(voucher.tokenId > 0, "Token ID must be greater than 0"); bytes32 hash = \_hash(voucher); //string memory hash=""; return ECDSA.recover(hash, signature); }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
but the result of this is not matching with actual signer. i think I am making some mistake in the hash function above used.  
  
0xe8c795f9168269940b31a470ad82e89a453e88b9 signer0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266 owner  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
below is the hash function.  
  
function \_hash(NFTVoucher calldata voucher) internal view returns (bytes32) { return \_hashTypedDataV4( keccak256( abi.encode( keccak256( "Nft(string URI,uint256 price)" ), keccak256(bytes(voucher.URI)), voucher.price ) ) ); }  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
reference to above \_hashTypedDataV4

1 **Answer**

**A1:**this is erc20permit example,I hope it can help you  
  
function permit(address owner, address spender, uint256 amount, uint256 deadline, uint8 v, bytes32 r, bytes32 s) public virtual override { require(deadline >= block.timestamp, "ERC20Permit: expired deadline"); bytes32 hashStruct = keccak256( abi.encode( keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)"), owner, spender, amount, nonce[owner], deadline ) ); bytes32 hash = keccak256( abi.encodePacked( '\x19\x01', keccak256(abi.encode( keccak256("EIP712Domain(string name,string version,uint256 chainId,address verifyingContract)"), keccak256(bytes(name\_)), keccak256(bytes(version())), chainId, address(this) ), hashStruct ) ); address signer = ecrecover(hash, v, r, s); require( signer != address(0) && signer == owner, "ERC20Permit: invalid signature" ); nonces[owner]++; \_approve(owner, spender, amount);}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]