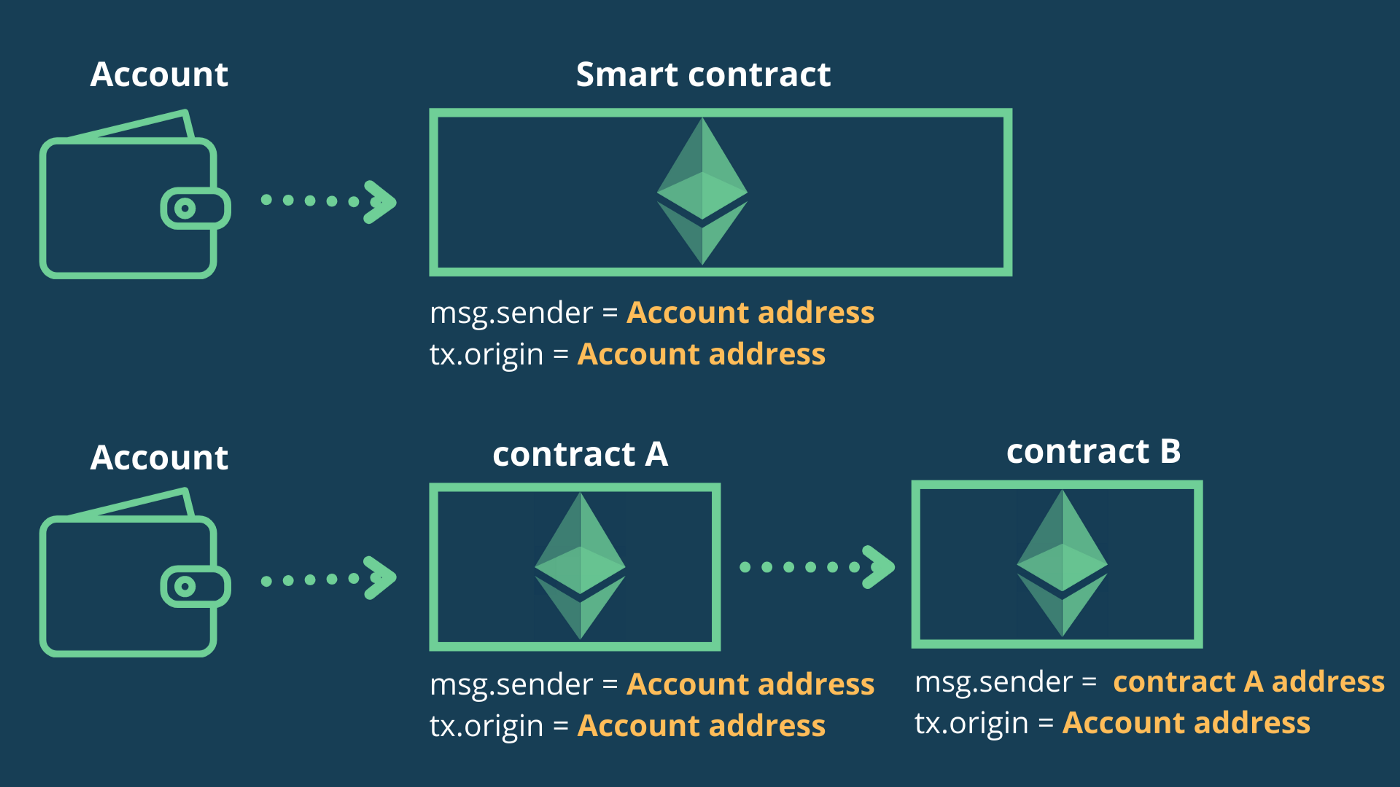
**tx.origin 与msg.sender**

它们都是solidity 中构建全局变量。根据solidity[文档](https://docs.soliditylang.org/en/v0.8.7/units-and-global-variables.html" \t "/Users/ansonxiang/Documents\\x/_blank)，tx.origin 保存交易发送者的地址，msg.sender 保存消息发送者的地址。

****msg.sender：****指直接调用智能合约函数的账户或智能合约地址。

****tx.origin：****指调用智能合约函数的账户地址，只有账户地址可以是tx.origin。



您可能会注意到，账户地址和智能合约地址都可以是 msg.sender 但 tx.origin 将始终是账户/钱包地址。

**强烈建议始终使用 msg.sender 进行授权或检查调用智能合约的地址。并且永远不要使用 tx.origin 进行授权，因为这可能会使合约容易受到网络钓鱼攻击。**

**tx.origin 噩梦**

Solidity 有一个全局变量，tx.origin其中包含最初发送调用（或交易）的账户地址。在 Solidity 的旧时代，这被用作确保只有 EOA（外部拥有的帐户）可以调用某些功能的一种方式。那么它与 有何不同msg.sender？让我们考虑以下场景。

如果账户 A 调用合约 B，B 调用合约 C，则 C 中msg.sender是合约 B，tx.origin是账户 A。如果这有助于保护某些功能或用于身份验证，为什么不建议再使用它？**有两个原因**。

**1、网络钓鱼攻击**

使用该tx.origin变量授权用户的合约通常容易受到网络钓鱼攻击，这些攻击可以诱骗用户对易受攻击的合约执行经过身份验证的操作。让我们考虑这个取自solidity docs的例子。

Alice 部署以下钱包。只有她可以调用transferTo函数。

// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

// THIS CONTRACT CONTAINS A BUG - DO NOT USE

contract TxUserWallet {

address owner;

constructor() {

owner = msg.sender;

}

function transferTo(address payable dest, uint amount) public {

require(tx.origin == owner);

dest.transfer(amount);

}

}

但是 Bob 部署了以下代码并欺骗 Alice 将以太币从她的钱包发送到这个合约地址。

// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

interface TxUserWallet {

function transferTo(address payable dest, uint amount) external;

}

contract TxAttackWallet {

address payable owner;

constructor() {

owner = payable(msg.sender);

}

receive() external payable {

TxUserWallet(msg.sender).transferTo(owner, msg.sender.balance);

}

}

当这个合约收到 Ether 时，receive()函数会自动运行，它会欺骗 AlicetransferTo从她自己的钱包合约中运行并窃取所有 Alice Ether。

由于tx.origin身份验证检查，它确实有效。tx.origin在这种情况下是 Alice 的地址，msg.sender也是 Alice 的钱包。Bob 对 Alice 进行了成功的网络钓鱼攻击。

这是众所周知的攻击向量，但遗憾的是 $RUNE 仍在使用tx.origin for approve and call方法。

**2、EIP-3074**

不应该使用 tx.origin 的另一个原因是今年晚些时候即将到来的上海硬分叉以及将包含在其中的 EIP-3074。EIP-3074引入了两个新的 EVM 指令 AUTH 和 AUTHCALL。第一个设置基于 ECDSA 签名授权的上下文变量。第二个作为授权发送呼叫。这实质上将 EOA 的控制权委托给了智能合约。这意味着智能合约将有一种方法可以在外部拥有账户的上下文中发送交易，从而绕过 tx.origin 检查。

EIP-3074 带来了其他好处，例如使用 ERC20 代币支付交易的可能性，因为智能合约将能够在外部拥有账户的情况下发送交易。有关更多信息，我建议阅读此博客文章。

如果您认为使用tx.origin身份验证是一个好主意，请三思而后行，一遍又一遍地阅读这篇文章。我不建议tx.origin在代码中的任何地方使用，如果您当前的系统正在使用这个全局变量，请停止一切​​并重构您的代码。您不想以新闻作为下一个协议而告终。