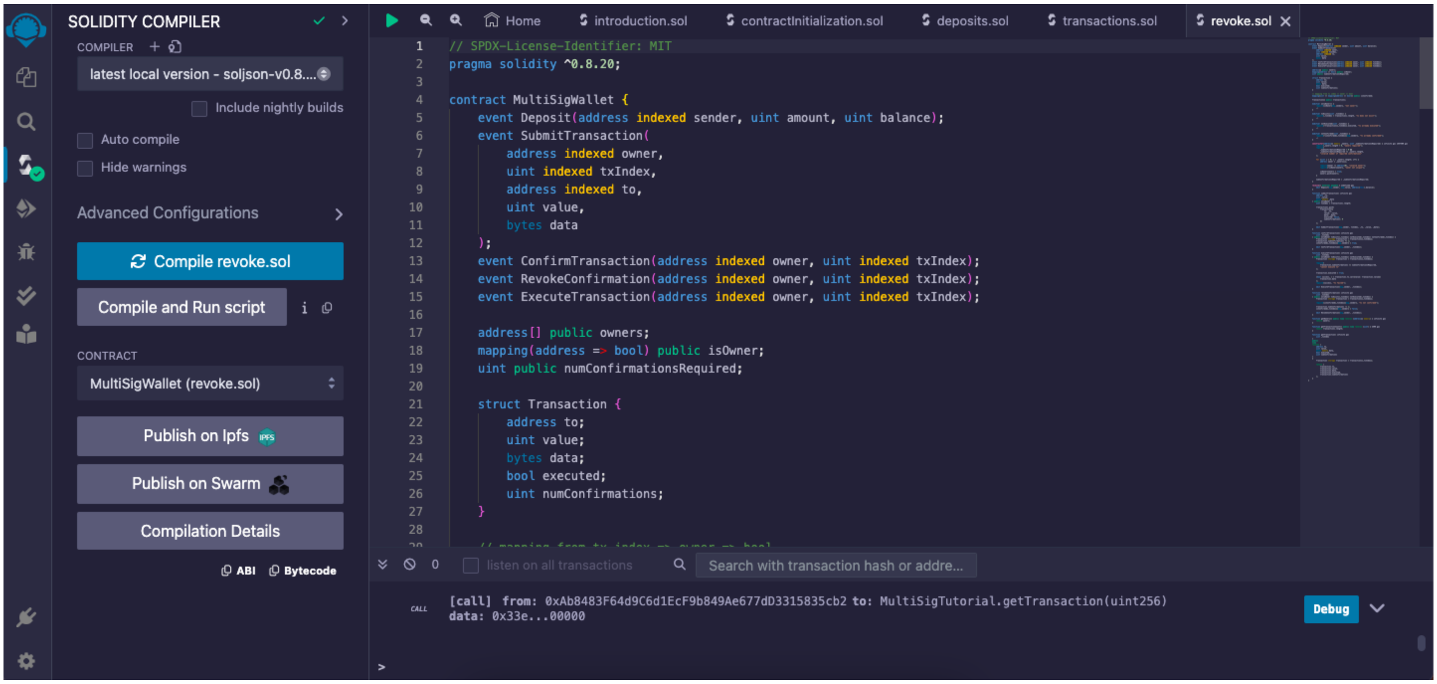
MultiSig Tutorial Documentation

In this documentation, I will take you through the steps that allowed me to explore the process of creating a multi-signature wallet, initialize the contract, deposit Ether, and submit, confirm, revoke, and execute transactions.

1. Compiling



SOLIDITY COMPILER

COMPILER: latest local version - soljson-v0.8....

□ Include nightly builds

□ Auto compile

□ Hide warnings

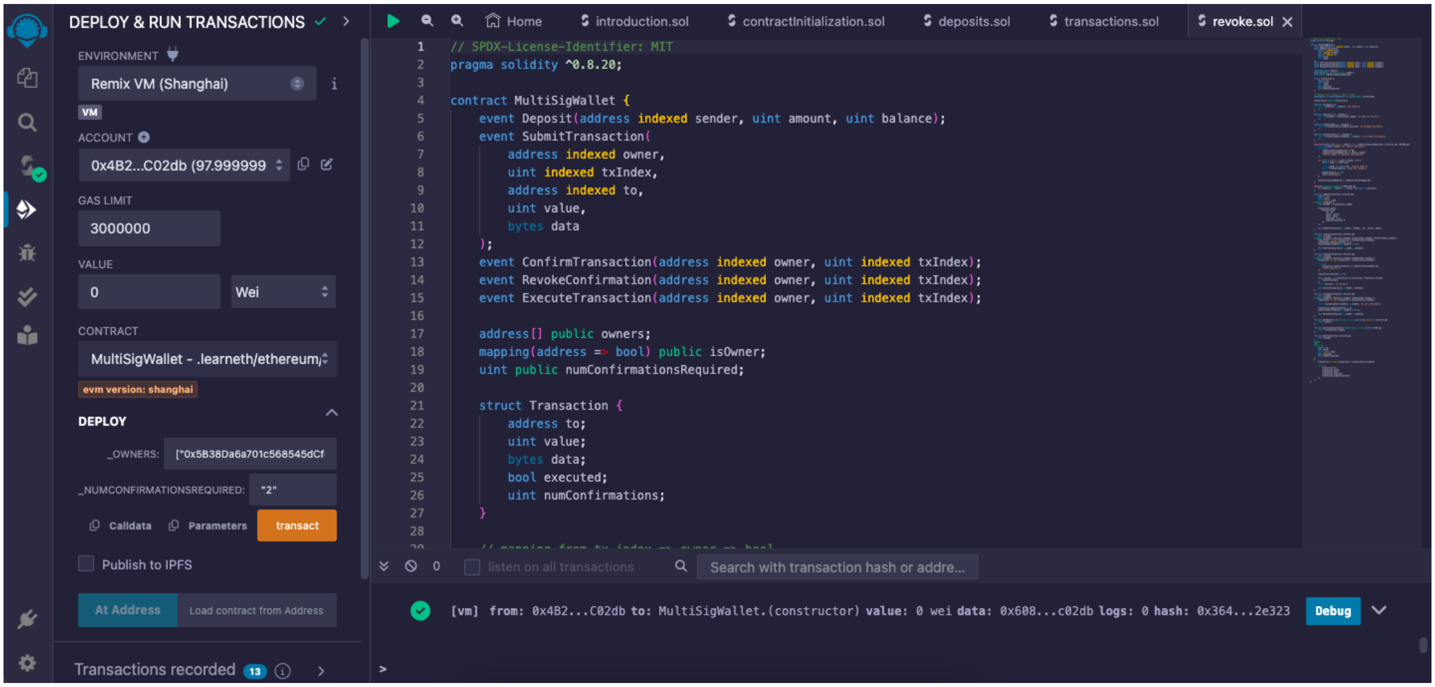
CONTRACT: MultiSigWallet (revoke.sol)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract MultiSigWallet {
    event Deposit(address indexed sender, uint amount, uint balance);
    event SubmitTransaction(
        address indexed owner,
        uint indexed txIndex,
        address indexed to,
        uint value,
        bytes data
    );
    event ConfirmTransaction(address indexed owner, uint indexed txIndex);
    event RevokeConfirmation(address indexed owner, uint indexed txIndex);
    event ExecuteTransaction(address indexed owner, uint indexed txIndex);

    address[] public owners;
    mapping(address => bool) public isOwner;
    uint public numConfirmationsRequired;

    struct Transaction {
        address to;
        uint value;
        bytes data;
        bool executed;
        uint numConfirmations;
    }

    // mapping from tx index => owner => bool
```

```
[call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: MultiSigTutorial.getTransaction(uint256)
data: 0x33e...00000
```

Here you can see that the contract has been successfully compiled. Compiling success can be verified because the contract is now available in the dropdown menu underneath the Contract section. The publishing options are also now available.

2. Deployment with Multiple Owners and Number of Confirmations Required



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Remix VM (Shanghai)

ACCOUNT: 0x4B2...C02db (97.999999

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: MultiSigWallet - .learneth/ethereum/

evm version: shanghai

DEPLOY

_OWNERS: ["0x5B38Da6a701c568545dCf

_NUMCONFIRMATIONSREQUIRED: "2"

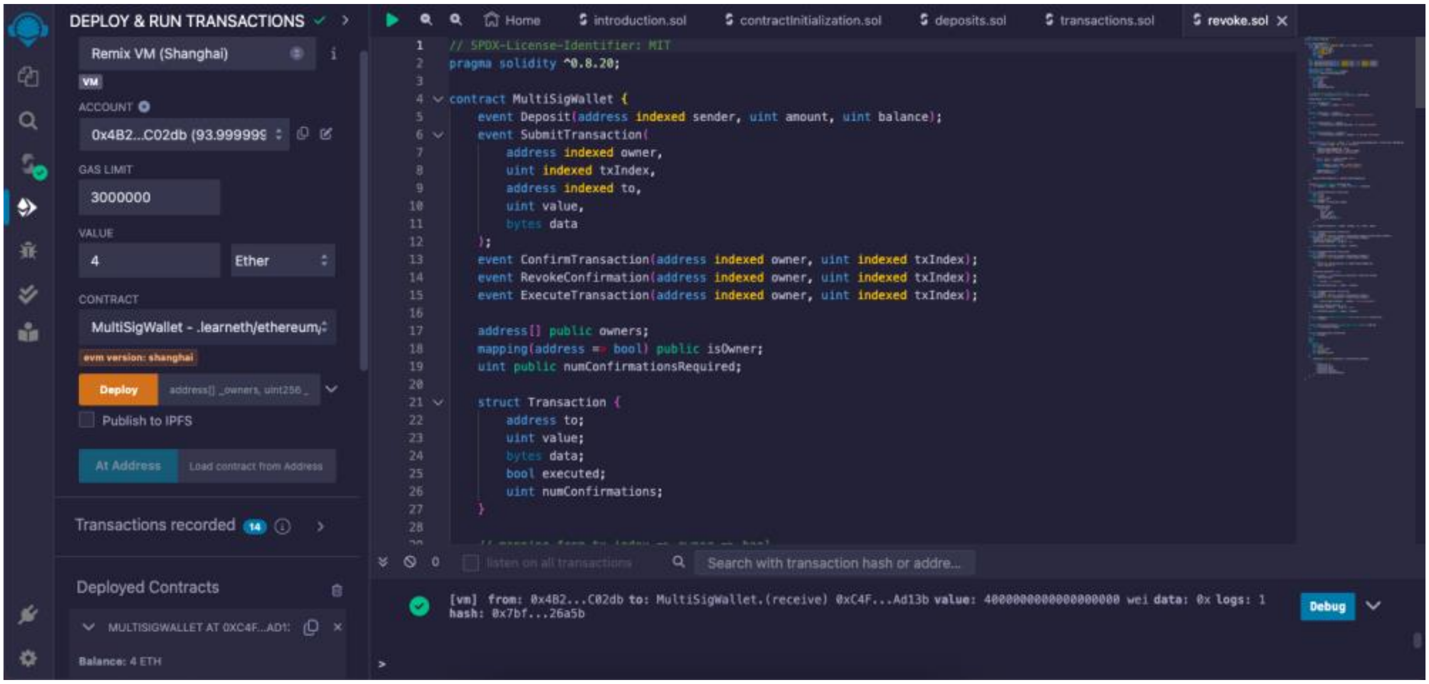
□ Publish to IPFS

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract MultiSigWallet {
    event Deposit(address indexed sender, uint amount, uint balance);
    event SubmitTransaction(
        address indexed owner,
        uint indexed txIndex,
        address indexed to,
        uint value,
        bytes data
    );
    event ConfirmTransaction(address indexed owner, uint indexed txIndex);
    event RevokeConfirmation(address indexed owner, uint indexed txIndex);
    event ExecuteTransaction(address indexed owner, uint indexed txIndex);

    address[] public owners;
    mapping(address => bool) public isOwner;
    uint public numConfirmationsRequired;

    struct Transaction {
        address to;
        uint value;
        bytes data;
        bool executed;
        uint numConfirmations;
    }

    // mapping from tx index => owner => bool
```

```
[vm] from: 0x4B2...C02db to: MultiSigWallet.(constructor) value: 0 wei data: 0x608...c02db logs: 0 hash: 0x364...2e323
```

Here I have deployed the contract with an array of three unique owners. I have also set the number of transaction confirmations required to 2 to ensure that no singular owner can confirm and execute their own transaction. Deployment can be verified by the green checkmark present in the log.

3. Funding the Wallet


DEPLOY & RUN TRANSACTIONS
Home
introduction.sol
contractInitialization.sol
deposits.sol
transactions.sol
revoke.sol
VM
evm version: shanghai
Deploy
address[] _owners, uint256 _
Publish to IPFS
At Address
Load contract from Address
listen on all transactions
Search with transaction hash or addre...
Debug

Remix VM (Shanghai)

ACCOUNT: 0x4B2...C02db (93.999999

GAS LIMIT: 3000000

VALUE: 4 Ether

CONTRACT: MultiSigWallet - .learneth/ethereum/

Transactions recorded 14

Deployed Contracts

MULTISIGWALLET AT 0XC4F...AD1:

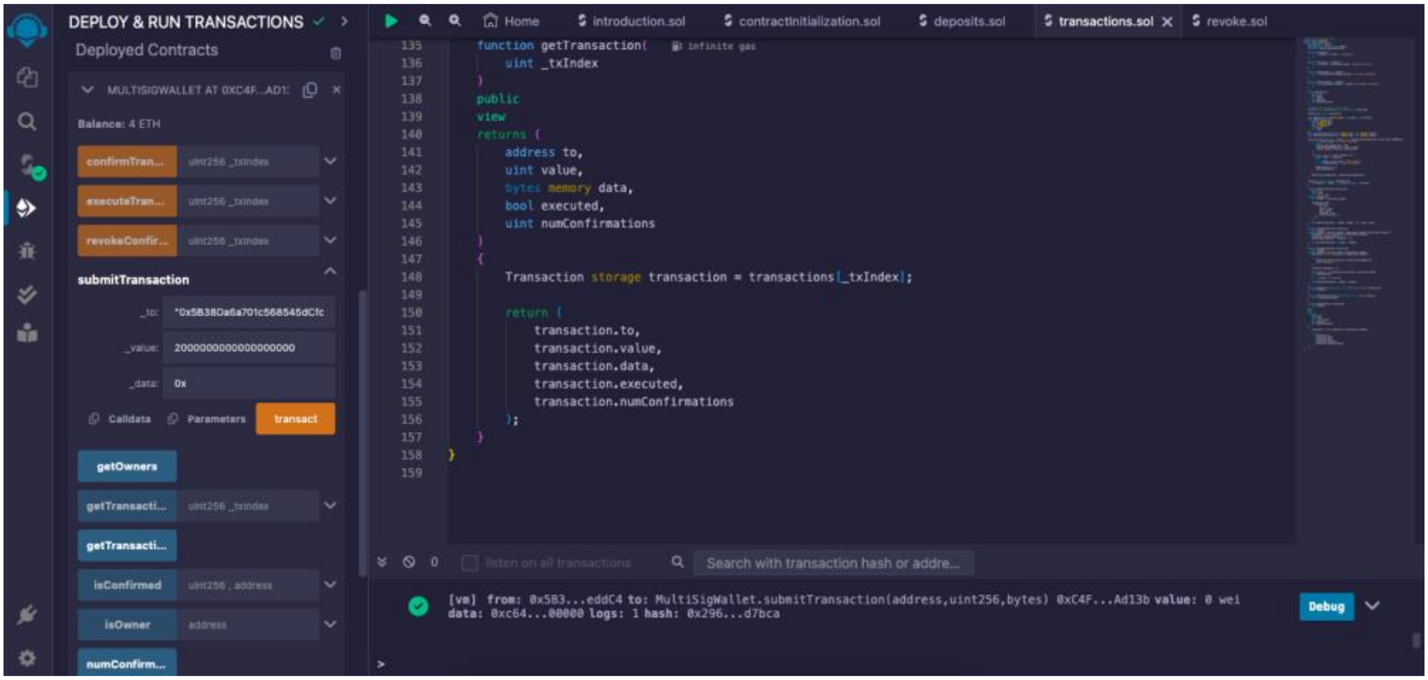
Balance: 4 ETH

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract MultiSigWallet {
    event Deposit(address indexed sender, uint amount, uint balance);
    event SubmitTransaction(
        address indexed owner,
        uint indexed txIndex,
        address indexed to,
        uint value,
        bytes data
    );
    event ConfirmTransaction(address indexed owner, uint indexed txIndex);
    event RevokeConfirmation(address indexed owner, uint indexed txIndex);
    event ExecuteTransaction(address indexed owner, uint indexed txIndex);

    address[] public owners;
    mapping(address => bool) public isOwner;
    uint public numConfirmationsRequired;

    struct Transaction {
        address to;
        uint value;
        bytes data;
        bool executed;
        uint numConfirmations;
    }
```

```
[vm] from: 0x4B2...C02db to: MultiSigWallet.(receive) 0xC4F...Ad13b value: 4000000000000000000 wei data: 0x logs: 1
hash: 0x7bf...26a5b
```

This picture shows that I have funded the wallet with 4 Ether from the specified account. I did this by choosing an account that had enough funds in it, filling out the Value section with the amount that I wanted to fund the wallet with, then I clicked the Transact button (not shown). Funding success can be verified by the green checkmark shown in the logs and the 4 ETH shown as the balance of the contract at the bottom of the Deployed Contracts section.

4. Submitting Eth Transfer Transaction



Deployed Contracts

MULTISIGWALLET AT 0XC4F...AD1:

Balance: 4 ETH

submitTransaction

_to: "0x5B38Da6a701c568545dCfc

_value: 2000000000000000000

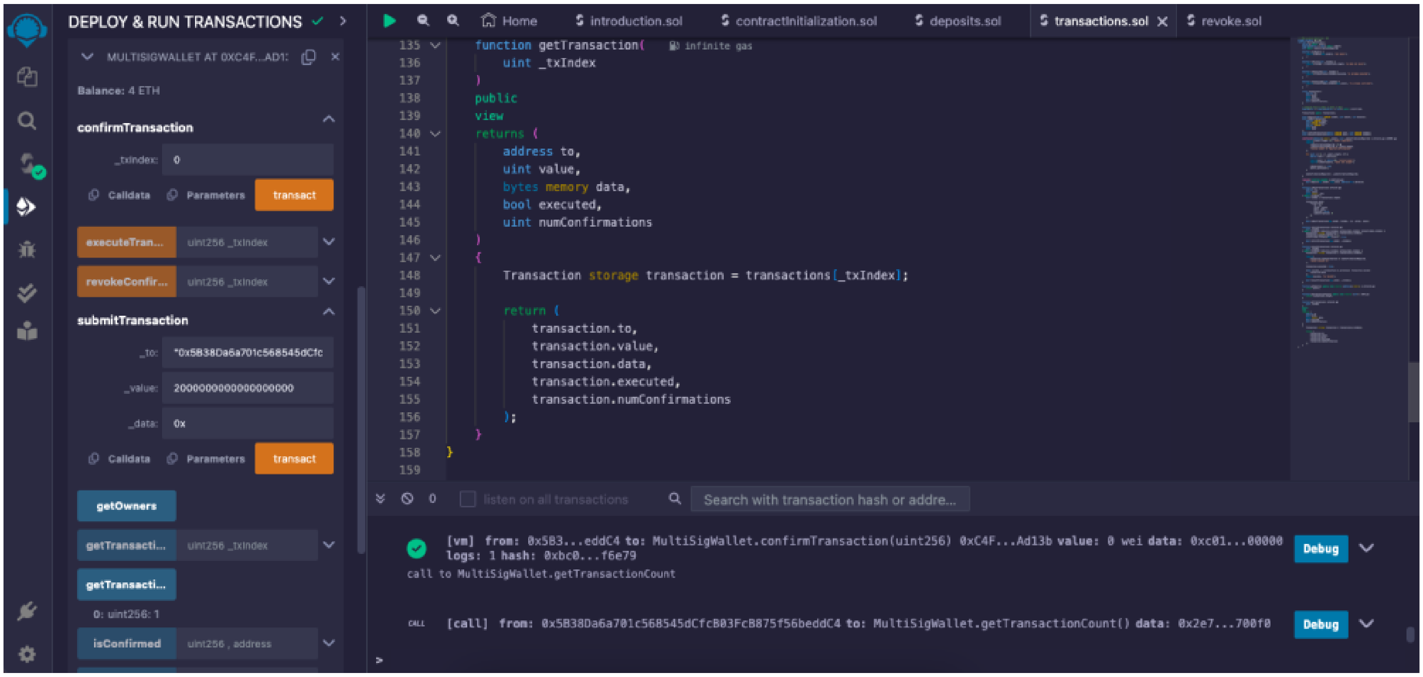
_data: 0x

```
    function getTransaction(
        uint _txIndex
    )
    public
    view
    returns (
        address to,
        uint value,
        bytes memory data,
        bool executed,
        uint numConfirmations
    )
    {
        Transaction storage transaction = transactions[_txIndex];

        return (
            transaction.to,
            transaction.value,
            transaction.data,
            transaction.executed,
            transaction.numConfirmations
        );
    }
}
```

```
[vm] from: 0x5B3...eddC4 to: MultiSigWallet.submitTransaction(address,uint256,bytes) 0xC4F...Ad13b value: 0 wei
data: 0xc64...00000 logs: 1 hash: 0x296...d7bca
```

In this screenshot, you can see that I submitted a transaction to send 2 Eth from one owner’s account to a different owner’s account. This was completed by filling out the appropriate fields under the submitTransaction function and pressing the transact button. Success verification can be seen by the green checkmark in the log.

5. Transaction Confirmations



DEPLOY & RUN TRANSACTIONS

MULTISIGWALLET AT 0XC4F...AD1:

Balance: 4 ETH

confirmTransaction
_txIndex: 0

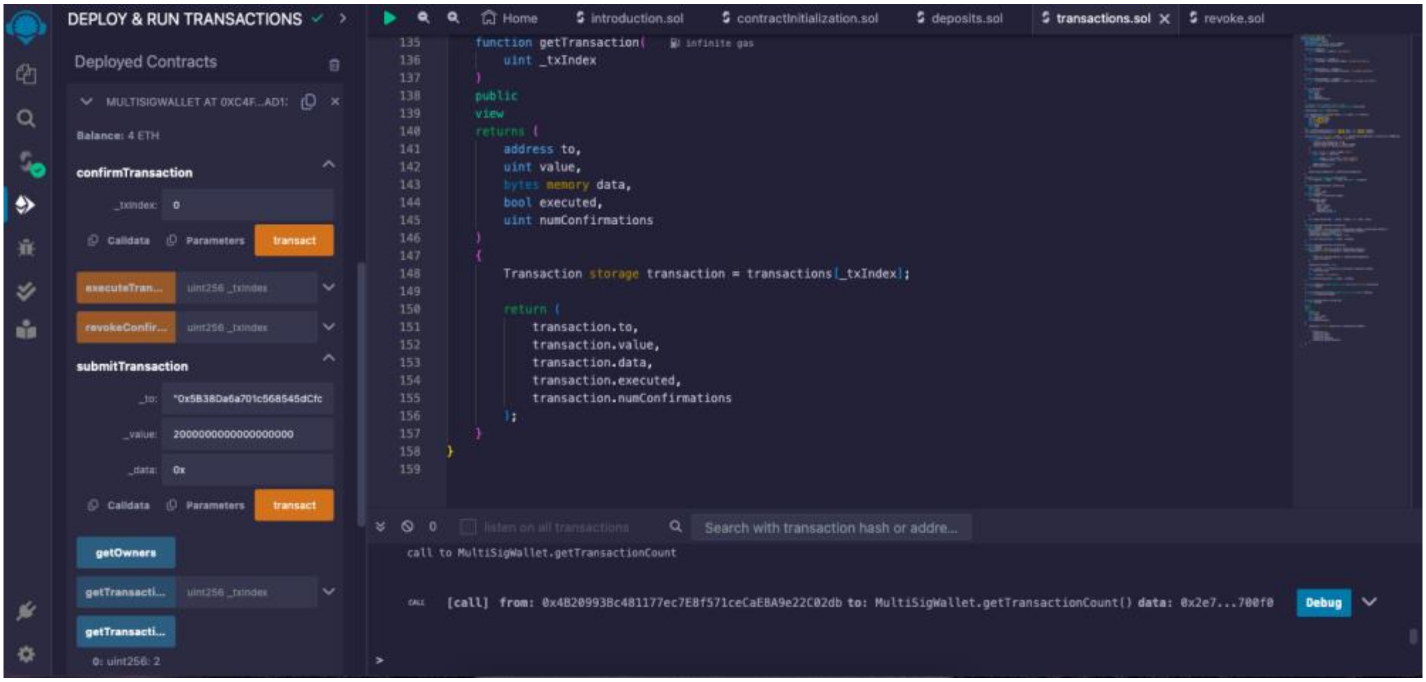
submitTransaction
_to: "0x5B38Da6a701c568545dCfc
_value: 2000000000000000000
_data: 0x

getTransacti...
0: uint256: 1

```
function getTransaction(
    uint _txIndex
)
public
view
returns (
    address to,
    uint value,
    bytes memory data,
    bool executed,
    uint numConfirmations
)
{
    Transaction storage transaction = transactions[_txIndex];

    return (
        transaction.to,
        transaction.value,
        transaction.data,
        transaction.executed,
        transaction.numConfirmations
    );
}
}
```

```
[vm] from: 0x5B3...eddC4 to: MultiSigWallet.confirmTransaction(uint256) 0xC4F...Ad13b value: 0 wei data: 0xc01...00000
logs: 1 hash: 0xbc0...f6e79
call to MultiSigWallet.getTransactionCount

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MultiSigWallet.getTransactionCount() data: 0x2e7...700f0
```

For this step, I remained on the same owner account, and I entered “0” into the transaction index field under the confirmTransaction function. I then pressed the transact button. After this, I pressed the getTransaction button and it returned a value of 1 meaning that the transaction has been confirmed one time.



DEPLOY & RUN TRANSACTIONS

Deployed Contracts

MULTISIGWALLET AT 0XC4F...AD1:

Balance: 4 ETH

confirmTransaction
_txIndex: 0

submitTransaction
_to: "0x5B38Da6a701c568545dCfc
_value: 2000000000000000000
_data: 0x

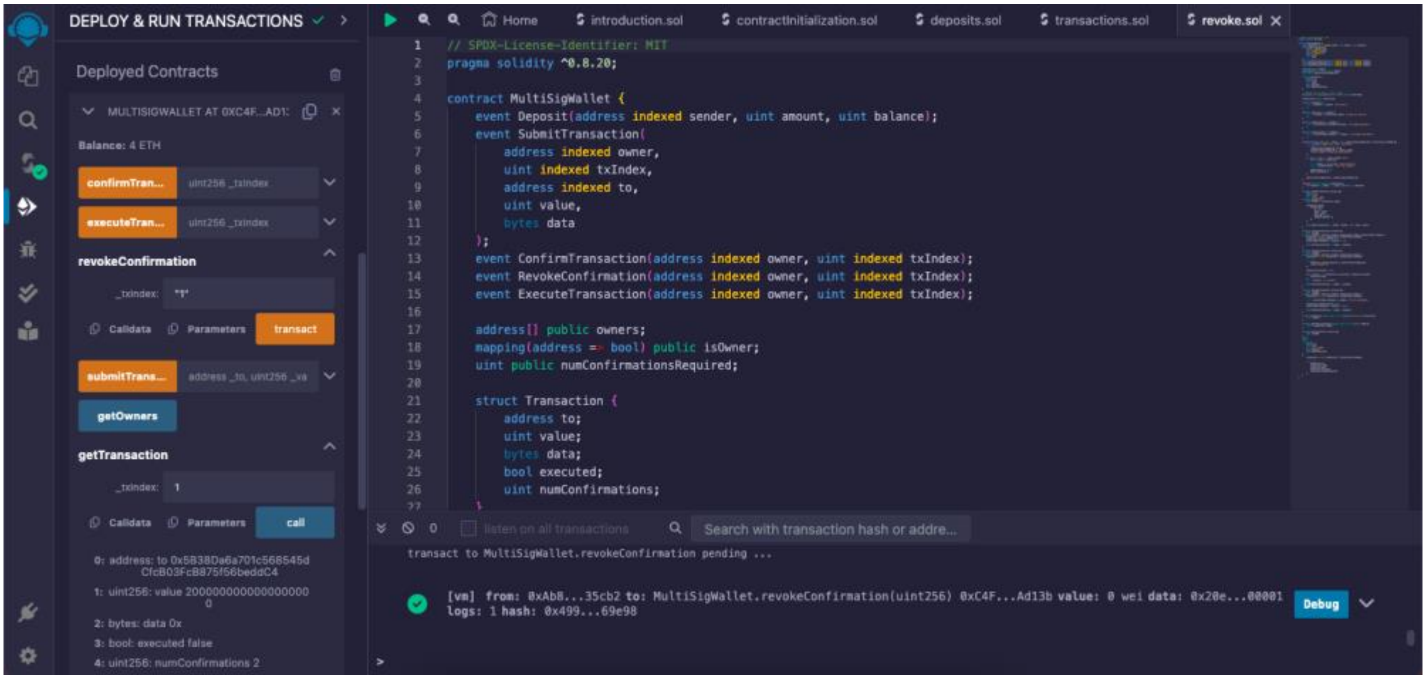
getTransacti...
0: uint256: 2

```
function getTransaction(
    uint _txIndex
)
public
view
returns (
    address to,
    uint value,
    bytes memory data,
    bool executed,
    uint numConfirmations
)
{
    Transaction storage transaction = transactions[_txIndex];

    return (
        transaction.to,
        transaction.value,
        transaction.data,
        transaction.executed,
        transaction.numConfirmations
    );
}
}
```

```
call to MultiSigWallet.getTransactionCount

CALL [call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: MultiSigWallet.getTransactionCount() data: 0x2e7...700f0
```

I then switched to a different owner’s account and repeated the process. You can see that the getTransaction value is now 2 meaning that the transaction has been confirmed two times, which is the number of confirmations required to execute a transaction in this contract.

6. Revoking a Confirmation



Balance: 4 ETH

revokeConfirmation _txIndex: "1"

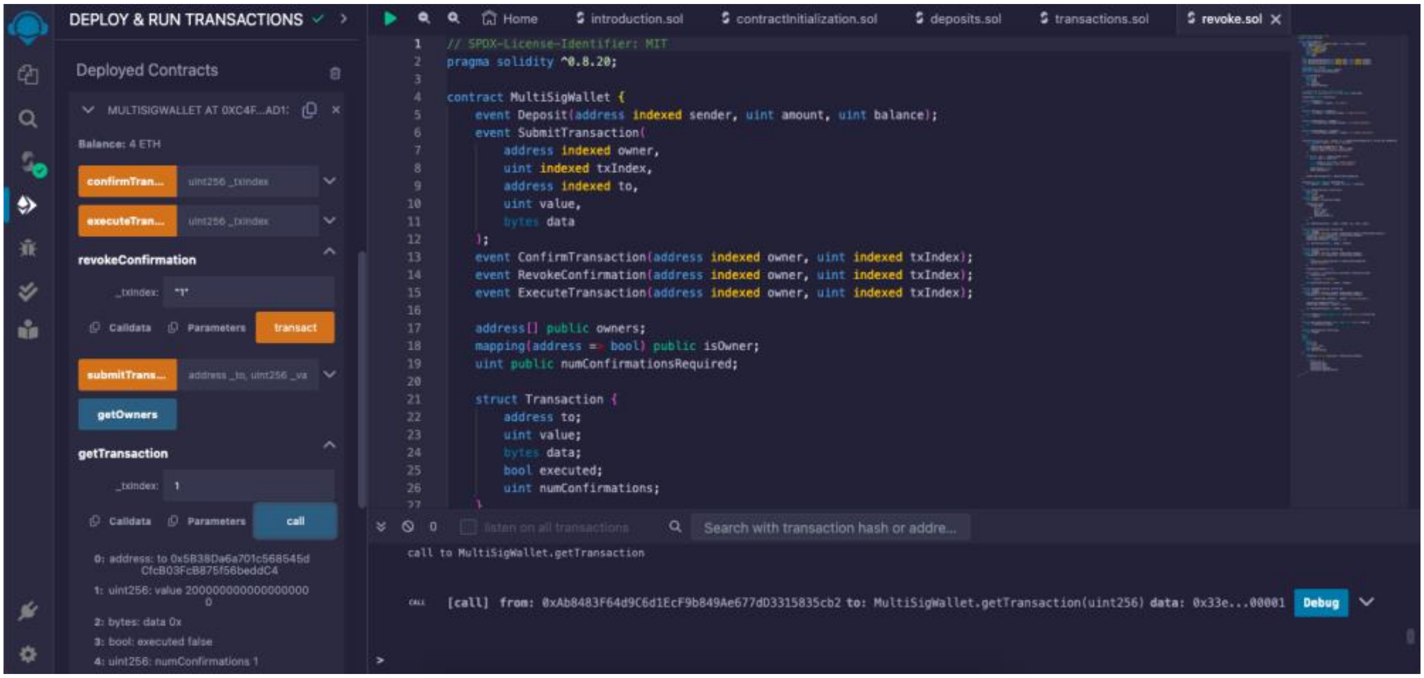
getTransaction _txIndex: 1

```
0: address: to 0x5B38Da6a701c568545d
       CfcB03FcB875f56beddC4
1: uint256: value 2000000000000000000
                 0
2: bytes: data 0x
3: bool: executed false
4: uint256: numConfirmations 2
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract MultiSigWallet {
    event Deposit(address indexed sender, uint amount, uint balance);
    event SubmitTransaction(
        address indexed owner,
        uint indexed txIndex,
        address indexed to,
        uint value,
        bytes data
    );
    event ConfirmTransaction(address indexed owner, uint indexed txIndex);
    event RevokeConfirmation(address indexed owner, uint indexed txIndex);
    event ExecuteTransaction(address indexed owner, uint indexed txIndex);

    address[] public owners;
    mapping(address => bool) public isOwner;
    uint public numConfirmationsRequired;

    struct Transaction {
        address to;
        uint value;
        bytes data;
        bool executed;
        uint numConfirmations;
    }
```

```
transact to MultiSigWallet.revokeConfirmation pending ...

[vm] from: 0xAb8...35cb2 to: MultiSigWallet.revokeConfirmation(uint256) 0xC4F...Ad13b value: 0 wei data: 0x20e...00001
logs: 1 hash: 0x499...69e98
```

I ran into some issues and ended up having to submit a second transaction. I confirmed that transaction 2 times to fulfill the requirement. Then I practiced revoking a confirmation. I chose an account, entered “1” into the transaction index field under the revokeConfirmation function (to indicate that I wanted to revoke the second transaction in the contract since I had to submit a new one), and then I pressed the transact button. Revocation submission can be verified by the green checkmark in the box.



Balance: 4 ETH

revokeConfirmation _txIndex: "1"

getTransaction _txIndex: 1

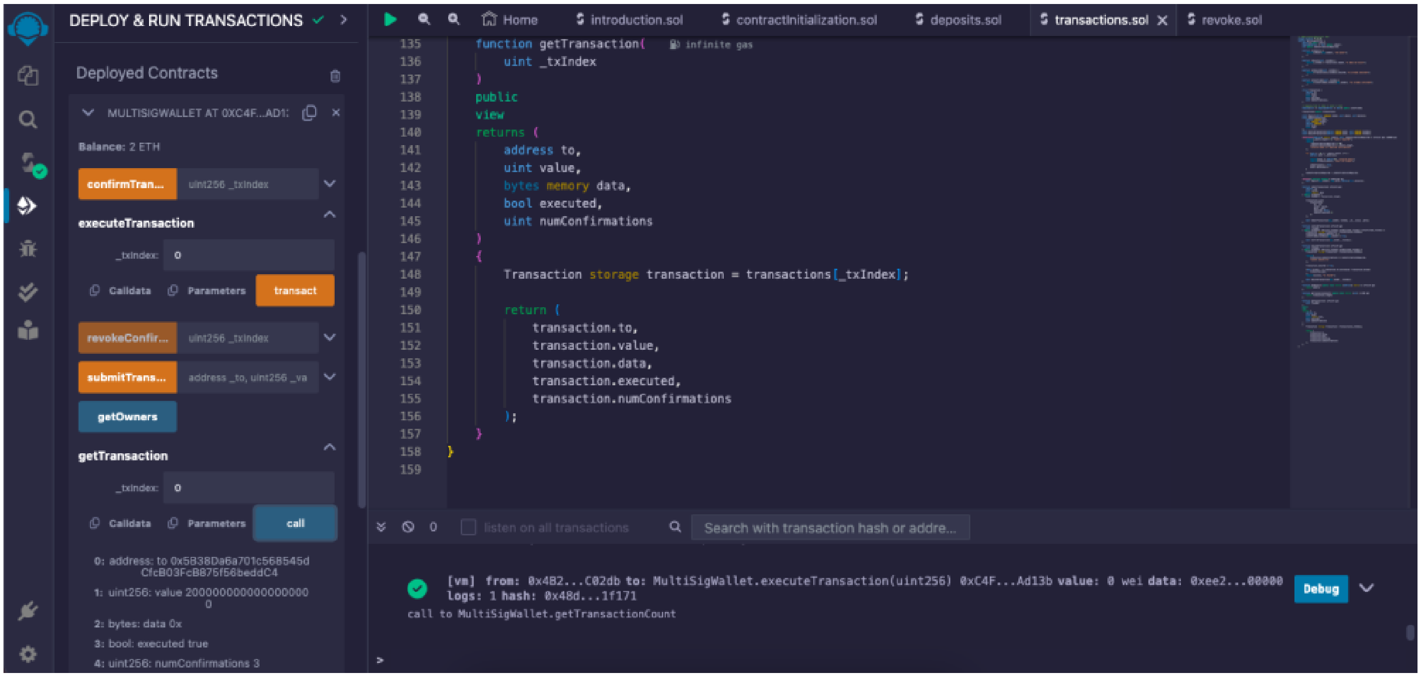
```
0: address: to 0x5B38Da6a701c568545d
       CfcB03FcB875f56beddC4
1: uint256: value 2000000000000000000
                 0
2: bytes: data 0x
3: bool: executed false
4: uint256: numConfirmations 1
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract MultiSigWallet {
    event Deposit(address indexed sender, uint amount, uint balance);
    event SubmitTransaction(
        address indexed owner,
        uint indexed txIndex,
        address indexed to,
        uint value,
        bytes data
    );
    event ConfirmTransaction(address indexed owner, uint indexed txIndex);
    event RevokeConfirmation(address indexed owner, uint indexed txIndex);
    event ExecuteTransaction(address indexed owner, uint indexed txIndex);

    address[] public owners;
    mapping(address => bool) public isOwner;
    uint public numConfirmationsRequired;

    struct Transaction {
        address to;
        uint value;
        bytes data;
        bool executed;
        uint numConfirmations;
    }
```

```
call to MultiSigWallet.getTransaction

CALL  [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: MultiSigWallet.getTransaction(uint256) data: 0x33e...00001
```

After I pressed the transact button and verified the revocation was successful, I entered “1” into the transaction index field of the getTransaction function and called the function. You can now see that it returned a Number of Confirmations value of 1 indicating that the confirmation was successfully revoked.

7. Executing a Transaction



Balance: 2 ETH

executeTransaction

_txIndex: 0

getTransaction

_txIndex: 0

```
0: address: to 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
1: uint256: value 2000000000000000000
2: bytes: data 0x
3: bool: executed true
4: uint256: numConfirmations 3
```

```
    function getTransaction(    infinite gas
        uint _txIndex
    )
    public
    view
    returns (
        address to,
        uint value,
        bytes memory data,
        bool executed,
        uint numConfirmations
    )
    {
        Transaction storage transaction = transactions[_txIndex];

        return (
            transaction.to,
            transaction.value,
            transaction.data,
            transaction.executed,
            transaction.numConfirmations
        );
    }
}
```

```
[vm] from: 0x4B2...C02db to: MultiSigWallet.executeTransaction(uint256) 0xC4F...Ad13b value: 0 wei data: 0xee2...00000
logs: 1 hash: 0x48d...1f171
call to MultiSigWallet.getTransactionCount
```

Finally, I reconfirmed the original transaction that I wanted to execute, then I entered "0" into the transaction index field of the executeTransaction function (to indicate that I wanted to execute my original transaction), and I pressed the transact button. Successful execution can be verified by the green checkmark in the log.