MultiSig Wallet

This is a MultiSig wallet, created in the Solidity language, which is used to write smart contracts to be deployed on the Ethereum blockchain network.

The purpose of a MultiSig wallet is to enhance security, by requiring the approval of a transaction by more than 50% of the owners of the wallet. The wallet, whose contract is included, enables the following functions:

1. Adding an Initial List of owners: The user is allowed to enter the addresses of the devices of the initial set of owners of the wallet. These owners are stored as an array of addresses and mapped to a boolean value, which indicates the status of their ownership.
2. Adding/Removing an owner: The users ( who are authorised to do so) are allowed to add or remove new owners from the list of owners. The list of owners is updated and the ownership status of the addresses are changed when this is called.
3. Submit a transaction: Every owner is allowed to propose and submit a transaction, if the transaction is deemed valid ( destination is valid).
4. Approve a transaction: Each user can use its address to approve a transaction. Each transaction’s status is ‘INVALID’ as a default, so it is not necessary to do it manually.
5. Confirm and execute a transaction: A final confirmation function is used to check if the transaction has been approved by at least 50% of the owners, and if so, it is executed.
6. Payable function: A payable no-name function is used to send ETH to the wallet.
7. Private Add transaction function: It is used to assign a transaction ID to every transaction that has been proposed by an owner and set its confirmation status as ‘INVALID’ by default.

Every transaction is an instance of the class ‘Transaction’ and contains the attributes of message(data), destination address, amount of ETH to be sent and the transaction ID.

Mappings are used to map:

1. Every owner to its ownership status(bool)
2. Every transaction to its confirmation status for a particular owner(bool)
3. Every transaction to a transaction ID.

The wallet can be improved by including private and public signatures, which will help in authenticating the owners.

A private signature is similar to a password, which is combined with the public key, and the message to generate a digital signature, which is used to sign any digital document. To verify this, the receiver needs to hash the original message and decrypt the digital signature with the help of the sender’s public key. If these two values match, then the message is verified.

The wallet’s functions can be tested by using Ganache, a test Ethereum network client and MyEtherWallet, which helps us deploy the contract on any Ethereum network.

(NOTE : Also included in the same repository is the Python code for a simple blockchain and cryptocurrency (BatCoin) that I created, that can be executed on Postman, a web app client)