



Experiment No.6
To design a smart contract implementing the concepts of wallet and transaction using solidity
Date of Performance:14/9/23
Date of Submission:



**AIM:** To design a smart contract implementing the concept of wallet and transaction using solidity

**Objective:** To develop a program using solidity to create a demonstration of wallet and perform transaction on the wallet **Theory:**

A blockchain wallet is a cryptocurrency wallet that allows users to manage different kinds of cryptocurrencies—for example, Bitcoin or Ethereum. A blockchain wallet helps someone exchange funds easily. Transactions are secure, as they are cryptographically signed. The wallet is accessible from web devices, including mobile ones, and the privacy and identity of the user are maintained. So a blockchain wallet provides all the features that are necessary for safe and secure transfers and exchanges of funds between different parties.

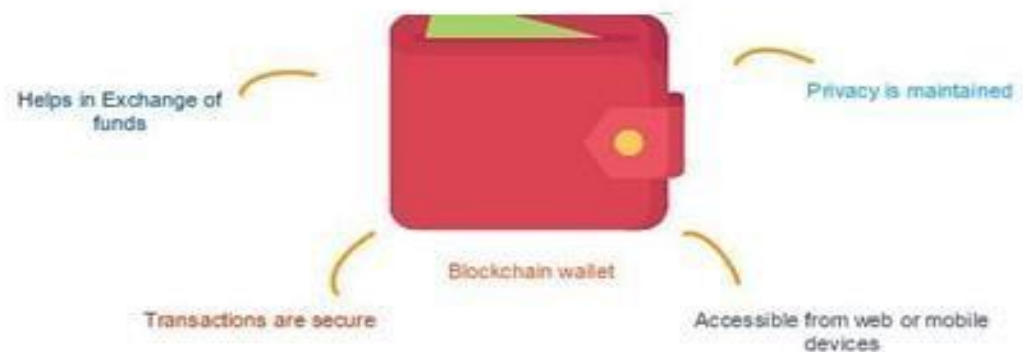


Fig. 6.1. Illustrative view of Wallet

### Blockchain Wallet Types

There are two types of blockchain wallets based on private keys: hot wallets and cold wallets. Hot wallets are like normal wallets that we carry for day-to-day transactions, and these wallets are user-friendly. Cold wallets are similar to a vault; they store cryptocurrencies with a high level of security.

#### *Hot Wallets and Cold Wallets*



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

---

Hot wallets are online wallets through which cryptocurrencies can be transferred quickly. They are available online. Examples are Coinbase and Blockchain.info. Cold wallets are digital offline wallets where the transactions are signed offline and then disclosed online. They are not maintained in the cloud on the internet; they are maintained offline to have high security. Examples of cold wallets are Trezor and Ledger.

With hot wallets, private keys are stored in the cloud for faster transfer. With cold wallets, private keys are stored in separate hardware that is not connected to the internet or the cloud, or they are stored on a paper document. Hot wallets are easy to access online 24/7 and can be accessed via a desktop or mobile device, but there is the risk of unrecoverable theft if hacked. With cold wallets, the method of the transaction helps in protecting the wallet from unauthorized access (hacking and other online vulnerabilities).

### Process:

Step 1. Open Remix **IDE** by typing URL <https://remix.ethereum.org/>.

Step 2. In Remix IDE select 'Solidity' plugins

Step 3. Click on File Explorer

Step 4. In the default workspace click on 'create new file'

Step 5. Give suitable name to the *file* with extension `.sol`

Step 6. Type the (Wallet and transaction) code in the editor section of the Remix IDE for newly created file (`.sol`)

Step 7. After typing the code for the smart contract in the newly creates `.sol` file, click on the compiler option available and then compile the file

Step 8. If no error, then click on the 'Deploy and Run' to execute the smart contract

### code:

```
// SPDX-License-Identifier: MIT    pragma
solidity ^0.8.0;
```



```
contract Transactions { address private owner;
    uint256 transactionCounts; mapping
    (address => uint) balanceOf; event
    Transfer(address indexed sender, address
    indexed receiver, uint256 amount,
    string remark, uint256 timestamp);
```

```
    struct TransferStruct {
        address sender;
        address receiver;
        uint256 amount;
        string remark; uint256
        timestamp;
    }
```

```
    TransferStruct[] transactions;
```

```
    constructor() { owner = msg.sender;
        balanceOf[tx.origin] = msg.sender.balance;
    }
```

```
    function getOwner() public view returns (address) { return owner;
    }
```

```
        function sendMoney(address payable receiver, uint256 amount, string memory
        remark) public returns(bool success) { if
        (balanceOf[owner] < amount) return false;
        balanceOf[owner] -= amount;
        balanceOf[receiver] += amount;
```

```
        transactionCounts += 1; transactions.push(
        TransferStruct(
            owner,
```



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

---

```
        receiver,
        amount,
        remark,
        block.timestamp
    p
)
);
    emit Transfer(msg.sender, receiver, amount, remark, block.timestamp); return true;
}
function getBalance(address addr) public view returns(uint) { return balanceOf[addr];
}
function getAllTransactions() public view returns(TransferStruct[] memory) { return
    transactions;
}
function  getTransactionsCount()  public  view  returns(uint256)  {    return
    transactionCounts;
}
}
```

### Conclusion:

In Solidity, a versatile programming language tailored for the creation of smart contracts on the Ethereum blockchain, diverse features are employed to facilitate and govern transactions, ensuring their security and reliability within decentralized applications. The incorporation of functionalities like modifiers and events is crucial for streamlining transactions within smart contracts. Modifiers empower developers to set predefined conditions on functions, guaranteeing that transactions adhere to specific requirements before execution. Simultaneously, events serve as a valuable mechanism to document and communicate critical transaction details to external parties, fostering transparency and enabling effective monitoring of contract interactions. Together, these features play a pivotal role in establishing transactions that are secure, trustworthy, and traceable within Ethereum-based smart contracts.