

1.1 Working with openSSL Tool (encrypt,decrypt text,generate private & public keys)

1.1) Working with openSSL Tool

What is openSSL? It is a cryptographic library that provides built-in commands to generate private & public keys, encrypts & decrypts given text, generates a digital signature, hash functions and also to fetch and IP addresses of a URL.

In Ubuntu, open Terminal and type the command below to check openSSL exists or not.
`openssl version`

1.1.1) Command to Encrypt the plaintext

AIM: To create an encrypted file using openssl command

PROCESS:

a) Open terminal and type below commands. It will create and open a new text file.

`touch plaintext.txt`

`open plaintext.txt`

b) Now add some text in the plaintext and save it.

c) In the terminal, type below openssl command. Set below passphrase.

`openssl enc -aes256 -salt -in plaintext.txt -out encrypted.enc`

Passphrase:12345678

1.1.2) Command to Decrypt the Ciphertext

AIM: To decrypt the plaintext from encrypted file using openssl command

PROCESS:

a) Open the terminal, type below openssl command. Set below passphrase.

Note: This passphrase should be same as for passphrase used for encrypted text in 1.1

`openssl enc -d -aes256 -in encrypted.enc -out decrypted.txt`

Passphrase:12345678

1.1.4) Command to generate private key

AIM: To generate a private key file (.pem) using openssl command

PROCESS:

a) Create a folder “BCLAB” and open it in a terminal

b) Type below openssl command which uses RSA algorithm to generate private key file. Enter the below passphrase.

`openssl genpkey -algorithm RSA -out private_key.pem -aes256`

Passphrase:1234

1.1.5) Command to generate public key using private key

AIM: To generate a public key file (.pem) using private key file

PROCESS:

a) Open the folder “BCLAB” in the terminal.

b) Check the file “private_key.pem” exists in folder or not. If not, create it using openssl command in 1.4

c) Type the below openssl command to generate the public key.

Enter the passphrase same as used for creating private key

`openssl rsa -in private_key.pem -pubout -out public_key.pem`

Passphrase:1234

1.2 Create a Digital Signature using private key for demo.txt

AIM: To generate a digital signature file using a private key.

PROCESS:

- a) Open folder “BCTLAB” in terminal
- b) Type below commands to create new text file “demo.txt”

touch demo.txt

open demo.txt

Add some content in the above file

c) Check the file **private_key.pem** file exists. If not, create one using the private key **openssl** command.

d) Type the below **openssl** command to create a digital signature. Enter passphrase **1234**

openssl dgst -sha256 -sign private_key.pem -out file.sig demo.txt

e) A new file “**file.sig**” will be generated

AIM: To generate a digital signature file using a private key.

PROCESS:

- a) Open folder “BCTLAB” in terminal
- b) Type below commands to create new text file “demo.txt”

touch demo.txt

open demo.txt

Add some content in the above file

c) Check the file **private_key.pem** file exists. If not, create one using the private key **openssl** command.

d) Type the below **openssl** command to create a digital signature. Enter passphrase **1234**

openssl dgst -sha256 -sign private_key.pem -out file.sig demo.txt

e) A new file “**file.sig**” will be generated

1.4 Node JS program to demonstrate encryption & decryption

AIM: To write and execute a Node JS program to demonstrate encryption and decryption

PROCESS:

- a) Create a folder “BCLAB” and open it in VS Code IDE
- b) Create a new file “Encrypt.js” , write below code and save it.
- c) Open Terminal and type below commands

npm install prompt-sync

node Encrypt.js

program :

```
const crypto = require("crypto");
const prompt=require("prompt-sync")()
const algorithm = "aes-256-cbc";
const key = crypto.randomBytes(32);
const iv = crypto.randomBytes(16);
const encrypt = (text) => {
    let cipher = crypto.createCipheriv(algorithm, key, iv);
    let encrypted = cipher.update(text, "utf8", "hex");
    let encrypted2=encrypted+cipher.final("hex");
    return encrypted2;
};
const decrypt = (encryptedText) => {
    let decipher = crypto.createDecipheriv(algorithm, key, iv);
    let decrypted = decipher.update(encryptedText, "hex", "utf8")
    let decrypted2=decrypted+decipher.final("utf-8")
    return decrypted2;
};
let plaintext = prompt("Enter some text:");
let ciphertext = encrypt(plaintext);
console.log("Given Plaintext:", plaintext);
console.log("Generated Ciphertext:", ciphertext);
console.log("Decrypted Text:", decrypt(ciphertext));
```

1.5 Node JS program to demonstrate digital signature

AIM: To write and execute a Node JS program to demonstrate digital signature

PROCESS:

- a) Create a folder “BCTLAB” and open it in VS Code IDE
- b) Create a new file “DigitalSignature.js” , write below code and save it.
- c) Open Terminal and type below commands

npm install prompt-sync

node DigitalSignature.js

code:

```
const prompt=require("prompt-sync")()
const { generateKeyPairSync, createSign, createVerify } = require("crypto");
const { publicKey, privateKey } = generateKeyPairSync("rsa",
{
    modulusLength: 2048,
});
let msg = prompt("Enter a message:");
const sign = createSign("SHA256");
sign.update(msg);
sign.end();
const signature = sign.sign(privateKey, "hex");
console.log("Digital Signature created for the message:"+msg)
msg = prompt("Enter message to verify:");
const verify = createVerify("SHA256");
verify.update(msg);
verify.end();
console.log("Signature Verified:", verify.verify(publicKey,signature, "hex"));
```

3.1 Solidity program to display owner address and message

AIM: to write a solidity program that display contract's owner address and a welcome message

PROCESS:

- a) Open REMIX IDE and create a new file message.sol
- b) Write the below solidity program and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and click on “compile:” button
- d) Goto Deploy & transaction tab, click on “Deploy” button
- e) Now invoke necessary functions and access variables used in the program.

Program:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Message
{
    string message;
    address owner=msg.sender;
    function setMessage(string memory _m) public
    {
        message=_m;
    }
    function getMessage() public view returns(string memory)
    {
        return message;
    }
    function getOwner() public view returns(address)
    {
        return owner;
    }
}
```

3.2 Solidity program to perform ether transfer

AIM: To write a solidity program that performs ether transfer

PROCESS:

- a) Open REMIX IDE and create a new file EtherTransfer.sol
- b) Write the below solidity program and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and click on “compile:” button
- d) Goto Deploy & transaction tab, click on “Deploy” button
- e) In the Accounts section ,select any account and copy it.This will be the receiver's account.
- f) Also Enter Amount in Ethers in Value section
- g) Now copy the receiver's account in the ‘sendEther’ function and call sendEther function.
- h) Amount will be transferred and call other functions to see outputs.

Program:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract EtherTransfer
{
    address public sender;
    address public receiver;
    uint public amount;
    function sendEther(address payable _receiver) public payable {
        require(msg.value > 0, "You must send some Ether");
        sender = msg.sender;
        receiver = _receiver;
        amount = msg.value;
        _receiver.transfer(msg.value);
    }
    function getDetails() public view returns (address, address, uint) {
        return (sender, receiver, amount);
    }
    function getReceiverBalance() public view returns (uint) {
        return address(receiver).balance;
    }
    function getSenderBalance() public view returns (uint) {
        return address(sender).balance;
    }
}
```

3.2 Solidity program to reverse a digit

AIM: To write a solidity program to reverse a digit

PROCESS:

- a) Open REMIX IDE and create a new file EtherTransfer.sol
- b) Write the below solidity program and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and click on “compile:” button
- d) Goto Deploy & transaction tab, click on “Deploy” button
- e) Enter a digit in the textbox beside getReverse function and call getReverse function.
- f) Output will be displayed

Program

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract DigitReverse
{
    function getReverse(uint num)public pure returns (uint)
    {
        uint rev=0;
        uint rem=0;
        while (num!=0)
        {
            rem=num%10;
            rev=rev*10+rem;
            num/=10;
        }
        return rev;
    }
}
```

3.4 Solidity program to Read and Check whether two numbers are Gapful or not

AIM: To write a solidity program to Read and Check whether two numbers are Gapful or not

PROCESS:

- a) Open REMIX IDE and create a new file Gapful.sol
- b) Write the below solidity program and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and click on “compile:” button
- d) Goto Deploy & transaction tab, click on “Deploy” button
- e) Enter a digit in the textbox beside checkGapful function and call checkGapful function.
- f) Output will be displayed

program:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Gapful {
    function isGapful(uint256 num)private pure returns (bool) {
        if (num < 100) {
            return false;
        }
        uint256 lastDigit = num % 10;
        uint256 firstDigit = num;
        while (firstDigit >= 10) {
            firstDigit /= 10;
        }
        uint256 divisor = firstDigit * 10 + lastDigit;
        return num % divisor == 0;
    }
    function display(uint256 num) public pure returns (string memory) {
        if (isGapful(num)) {
            return "Gapful";
        } else {
            return "Not Gapful";
        }
    }
}
```

3.5 Solidity program to demonstrate on Parametrized constructor

AIM: To write a solidity program to demonstrate on Parametrized constructor

PROCESS:

- a) Open REMIX IDE and create a new file ConstructorDemo.sol
- b) Write the below solidity program and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and click on “compile:” button
- d) Goto Deploy & transaction tab, Enter two numbers separated by commas in the textbox beside “Deploy” button.Click on the “Deploy” button.
- e) Call get() function.
- f) Output will be displayed

Program:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ConstructorDemo
{
    uint public x;
    uint public y;
    constructor(uint _x,uint _y)
    {
        x = _x;
        y = _y;
    }
    function get() public view returns(uint)
    {
        return x+y;
    }
}
```

3.6 Solidity program to Read an Array dynamically and find the sum of biggest and smallest elements

AIM: To write a solidity program to Read an Array dynamically and find the sum of biggest and smallest elements.

PROCESS:

- a) Open REMIX IDE and create a new file ArrayDemo.sol
- b) Write the below solidity program and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and click on “compile” button
- d) Goto Deploy & transaction tab, Enter a number in the textbox beside the “readArray” button.Click on the “readArray” button.
- e) Repeat above step-d for 5-6 times with different numbers.
- f) Call display,get,getLength functions.
- g) Output will be displayed

Program:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ArrayDemo
{
    uint[] arr;
    function readArray(uint _value) public
    {
        arr.push(_value);
    }
    function getLength() public view returns(uint)
    {
        return arr.length;
    }
    function get() public view returns(uint,uint,uint)
    {
        uint max=arr[0];
        uint min=arr[0];
        for(uint i=1;i<arr.length;i++) {
            if(arr[i]>max) {
                max=arr[i]; }
            if(arr[i]<min) {
                min=arr[i]; } }
        return (max,min,max+min); }
    function display()public view returns(uint[] memory)
    {
        return arr;
    }
}
```

3.7 Solidity program to compute Student Grade Report using Average marks using struct

AIM: To write a Solidity program to compute Student Grade Report using Average marks using struct

PROCESS:

- a) Open REMIX IDE and create a new file GradeReport.sol
- b) Write the below solidity program and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and click on “compile” button
- d) Goto Deploy & transaction tab, Enter a number in the textbox beside the “readArray” button. Click on the “readArray” button.
- e) Repeat above step-d for 5-6 times with different numbers.
- f) Call display,get,getLength functions.
- g) Output will be displayed

program:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract GradeReport
{
    struct student
    {
        string name;
        uint rollno;
        uint s1; uint s2; uint s3;
    }
    student[] public stds;
    function setStudent(string memory n,uint r,uint m1,uint m2,uint m3)public
    {
        stds.push(student(n,r,m1,m2,m3));
    }
    function getAverage(uint index)private view returns (uint)
    {
        uint total=stds[index].s1+stds[index].s2+stds[index].s3;
        return total/stds.length;
    }
    function getReport(uint index)public view returns (string memory) {
        uint report=getAverage(index);
        if (report>=70)
        {
            return "Distinction";
        } else if(report>=60 && report <70)
        {
            return "First class"; } else if(report>=50 && report <60) { return "Second
        class"; }
        else return "Fail"; }
    function getStudentCount()public view returns(uint) {
        return stds.length; } }
```

3.8 Solidity Program to write Self Ether Transfer using mapping

AIM: To write a Solidity Program to write Self Ether Transfer using mapping

PROCESS:

- a) Open REMIX IDE and create a new file SelfEtherTransfer.sol
- b) Write the below solidity program and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and click on “compile” button
- d) Goto Deploy & transaction tab,click on “Deploy” button.
- e) Enter a number in the textbox under the “Value” in Ethers.Click on the “deposit” button.
- f) Call withdraw,balance,getbalances functions.
- g) Output will be displayed

program:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract EthersMapping
{
    mapping ( address => uint ) public balances;
    function deposit() public payable
    {
        balances[msg.sender]+=msg.value;
    }
    function getBalance() public view returns (uint)
    {
        return balances[msg.sender];
    }
    function withdraw(uint _amount) public
    {
        require(balances[msg.sender]>=_amount, "Insufficient funds");
        balances[msg.sender]-=_amount;
        payable (msg.sender).transfer(_amount);
    }
}
```

3.9 Solidity program to import a Contract into another Contract

AIM: To write a Solidity program to import a Contract into another Contract

PROCESS:

- a) Open REMIX IDE and create new files: A.sol and B.sol
- b) Write the below solidity programs and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and compile A.sol and B.sol
- d) Goto Deploy & transaction tab, Select A.sol and click on “Deploy” button.
- e) Next Select B.sol, copy A.sol deployed contract address and paste it beside ‘Deploy’ button. Now click on the “Deploy” button.
- f) Call “showB()” function
- g) Output will be displayed

program:

A.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract A
{
    function showA()external pure returns(string memory)
    {
        return "Hello from A";
    }
}
```

B.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "./A.sol";
contract B
{
    A a1;
    constructor(address _a1)
    {
        a1 = A(_a1);
    }
    function showB()public view returns(string memory)
    {
        return a1.showA();
    }
}
```

3.10 Solidity program to demonstrate inheritance

AIM: To write a Solidity program to demonstrate inheritance

PROCESS:

- a) Open REMIX IDE and create new files: A.sol and B.sol
- b) Write the below solidity programs and save it
- c) Goto Solidity Compiler tab and choose the compiler 0.8.0 and compile A.sol and B.sol
- d) Goto Deploy & transaction tab,Select B.sol and click on “Deploy” button.
- e) Call “showB()” and “showA()” functions
- f) Output will be displayed

program:

A.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract A
{
    function showA()public pure returns(string memory)
    {
        return "Hello from A contract";
    }
}
```

B.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "./A.sol";
contract B is A
{
    function showB()public pure returns(string memory)
    {
        return "Hello from B Contract";
    }
}
```

4.3 Transfer NEO funds between NEO accounts

AIM: To perform fund transfer between NEO Wallet accounts

PROCESS:

- a. To transfer funds, first we should have wallet accounts. Ensure you have created at least two wallet accounts using above 4.2 task.
- b. Consider we have TWO user defined accounts namely “mvsr” and “IOT”. We already have a “genesis” account whose wallet status is shown below.
- c. To transfer funds between “mvsr” and “IOT”, we must first import a few NEO balances and NEO GAS from the “genesis” account. Under the Blockchains tab, right click on “neowallet.neo-express” and select “Transfer assets”.
- d. Choose type of asset as “NEO”
- e. Enter the amount of NEO to be transferred. For example 100.
- f. Choose the sender account as “genesis” and receiver account as “mvsr”. After a few seconds , funds will be transferred and balance will be updated in both accounts.
- g. Repeat step-c and Choose type of asset as “GAS”
- h. Repeat step-f and enter “GAS” amount as 1000. After few seconds “1000” GAS will be transferred from “genesis” to “mvsr”
- i. Now repeat step-c to step-f to transfer funds from “mvsr” to “IOT”. After transferring 50 NEO from “mvsr” to “IOT”, the status of accounts is shown below.

5.1 Linking Ganache in Rabby Wallet and perform Ether Transfer

AIM: To link Ganache in Rabby Wallet and perform Ether Transfer between Ganache accounts.

PROCESS:

- a. Open Ganache GUI Tool**
- b. Ensure Rabby Wallet is added as Chrome Extension in Google Chrome browser. Then**
Open Rabby Wallet Tool
- c. Create a new Wallet address in Rabby Wallet**
- d. After logging into Rabby Wallet, goto settings and Click on “Add custom network”**
- e. Fill the details to Add Custom Network as shown below and click on confirm.**
- f. In the Rabby wallet main page, click on your wallet account. Ie. Seed Phrase-1**
- g. A new page will appear and click on “Add new address”**
- h. Click on “Import Private Key”.**
- i. Copy Ganache First Account’s Private Key and paste it in “Import Private Key” text box. Click on confirm. Ganache First account will be linked in Rabby Wallet successfully. Change its name to “GanachePrivateKey-1”.**
- j. Similarly add Ganache second account like above steps and change its name to “GanachePrivateKey-2”**
- k. In the Rabby Wallet main page, Click on “send”. Choose Send account as “GanachePrivateKey-1” and To address as “GanachePrivateKey-2”.**
- l. Select amount as Custom Network “ETH” (100)**
- m. Now enter the amount to transfer and click on “Send”. Click on “sign” and “confirm” transaction.**
- n. After a few seconds, the fund transfer will be successful between two Ganache accounts.**
- o. Open Ganache and check the status of accounts and a block will be created to record the transaction.**

5.2 Linking Ganache in Remix IDE and execute a smart contract

AIM: To Link Ganache GUI in Remix IDE and run a smart contract.

PROCESS:

- a. Open Ganache GUI.
- b. Open Desktop Remix IDE and create a new file “Hello.sol”
- c. Write the below Solidity program and save it.
- d. In Remix IDE, change the solidity compiler version to 0.8.10
- e. Under Deploy & Run Transactions Tab, change Environment to “Custom - External http Provider”. A dialog box appears as shown below. Edit Ganache URL to <http://127.0.0.1:7545> and click ok.
- f. Now you will be connected to Ganache and you can access Ganache accounts in Remix IDE
- g. Click on Deploy and call solidity methods. The block will be created in Ganache Tool.

PROGRAM:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Hello
{
    string message="Hello World";
    function get()public view returns (string memory)
    {
        return message;
    }
}
```

5.3 Linking Ganache in Remix IDE and perform ether transfer using smart contract

AIM: To link Ganache in Remix IDE and perform ether transfer using smart contract

PROCESS:

- a. Open Ganache GUI.
- b. Open Desktop Remix IDE and create a new file “Transfer.sol”
- c. Write the below Solidity program and save it.
- d. In Remix IDE, change the solidity compiler version to 0.8.10
- e. Under Deploy & Run Transactions Tab, change Environment to “Custom - External http Provider”. A dialog box appears as shown below. Edit Ganache URL to <http://127.0.0.1:7545> and click ok.
- f. Now you will be connected to Ganache and you can access Ganache accounts in Remix IDE
- g. Click on Deploy and call solidity methods. The block will be created in Ganache Tool.

Program:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract EtherTransfer {
    address public sender;
    address public receiver;
    uint public amount;
    function sendEther(address payable _receiver) public payable {
        require(msg.value > 0, "You must send some Ether");
        sender = msg.sender;
        receiver = _receiver;
        amount = msg.value;
        _receiver.transfer(msg.value);
    }
    function getDetails() public view returns (address, address, uint) {
        return (sender, receiver, amount);
    }
    function getReceiverBalance() public view returns (uint) {
        return address(receiver).balance;
    }
    function getSenderBalance() public view returns (uint) {
        return address(sender).balance;
    }
}
```

5.4 Node js program that connects to Ganache, display Ganache accounts and their balances

AIM: To Write a NodeJS program that connects to Ganache, display Ganache accounts and their balances.

PROCESS:

- a. Open Ganache GUI
- b. Create a new folder “nodeganache” and open it in VS Code IDE
- c. Create a new file display.js and write the below program and save it.
- d. Open Command Prompt and goto the “nodeganache” directory
- e. Type below nodejs commands to run the program and get the output
npm install web3 prompt-sync
node display.js

PROGRAM:

```
const {Web3}=require("web3")
const prompt=require("prompt-sync")()
async function sendEther() {
    try {
        const web3=new Web3("http://127.0.0.1:7545")
        console.log("Connected to Ganache...")
        const accounts=await web3.eth.getAccounts()
        const sender=accounts[0]
        const receiver=accounts[1]
        console.log("Ganache Accounts are:\n====")
        for (let i = 0; i < accounts.length; i++)
        {
            console.log("Account-"+i+"="+accounts[i])
        }
        let a1=parseInt(prompt("Enter an account index to fetch balance:"))
        if(a1<0||a1>accounts.length)
            console.log("Invalid Index")
        else
        {
            console.log("Account:"+a1+"="+accounts[a1])
            const sbal=await web3.eth.getBalance(accounts[a1])
            let ebal=web3.utils.fromWei(sbal,"ether")
            console.log("====\nBalance of Account-"+a1+"="+ebal+" Ethers")
        }
    } catch (error)
    {
        console.log("Error in Connection="+error); } } sendEther()
```

5.5 Node js program that connects to Ganache and performs fund transfer among dynamic accounts.

AIM: To Write a NodeJS program that connects to Ganache and performs fund transfer among dynamic accounts.

PROCESS:

- a. Open Ganache GUI
- b. Create a new folder “nodeganache” and open it in VS Code IDE
- c. Create a new file transfer.js and write the below program and save it.
- d. Open Command Prompt and goto the “nodeganache” directory
- e. Type below nodejs commands to run the program and get the output

```
npm install web3 prompt-sync
```

```
node transfer.js
```

PROGRAM:

```
const {Web3}=require("web3")

const prompt=require("prompt-sync")()

async function sendEther()

{

try {

const web3=new Web3("http://127.0.0.1:7545")

console.log("Connected to Ganache...")

const accounts=await web3.eth.getAccounts()

console.log("Ganache Accounts are:\n=====")

for (let i = 0; i < accounts.length; i++)

{



console.log("Account-"+i+"="+accounts[i])

}

let s=parseInt(prompt("Enter index of sender account:"))

let r=parseInt(prompt("Enter index of receiver account:"))

if(s<0|s>accounts.length || r<0| r>accounts.length)
```

```

console.log("Invalid Sender/Receiver Index entered!")

else

{

let sbal,rbal,sebal,rebal;

console.log("Before Transfer..Account Status:\n===== ")

console.log("Sender Account:"+s+"="+accounts[s])

console.log("Receiver Account:"+r+"="+accounts[r])

sbal=await web3.eth.getBalance(accounts[s])

rbal=await web3.eth.getBalance(accounts[r])

sebal=web3.utils.fromWei(sbal,"ether")

rebal=web3.utils.fromWei(rbal,"ether")

console.log("=====\nBalance of Account-"+s+"="+sebal+" Ethers")

console.log("Balance of Account-"+r+"="+rebal+" Ethers")

let amt=parseInt(prompt("Enter amount to transfer:"))

const txn=await

web3.eth.sendTransaction({from:accounts[s],to:accounts[r],value:web3.utils.toWei(amt,"ether")})

console.log("Transaction success...Txn Hash="+txn.transactionHash)

console.log("===== \nAfter Transfer..Account Status:\n===== ")

console.log("Sender Account:"+s+"="+accounts[s])

console.log("Receiver Account:"+r+"="+accounts[r])

sbal=await web3.eth.getBalance(accounts[s])

rbal=await web3.eth.getBalance(accounts[r])

sebal=web3.utils.fromWei(sbal,"ether")

rebal=web3.utils.fromWei(rbal,"ether")

console.log("=====\nBalance of Account-"+s+"="+sebal+" Ethers")

console.log("Balance of Account-"+r+"="+rebal+" Ethers")

} } catch (error) { console.log("Error in Connection="+error) } } sendEther()

```