Bhartiya Vidya Bhavan's
Sardar Patel Institute of Technology, Mumbai-400058
Department of Electronics and Telecommunication Engineering
IT424: Blockchain Technology and Applications

Lab-7: Blockchain and Cybersecurity Part-I
Develop a blockchain application for Cybersecurity.

Name: Shubham Golwal **UID: 2020300015**

Objective: Develop a blockchain application for Cybersecurity

Outcomes: After successful completion of lab students should be able to
Implement an Ethereum private blockchain
Build two-factor authentication (2FA) using Blockchain
Write a smart contract using Solidity Language
Compile and run the 2FA using Ethereum Blockchain
Use REST API and Flask microframework

System Requirements:
PC (C2D, 8GB RAM, 100GB HDD space and NIC),Ubuntu Linux 14.04/20.04
Internet connectivity,Python Cryptography and Pycrypto,Nodejs, Truffle,Ganache-cli
, solidity,REST API

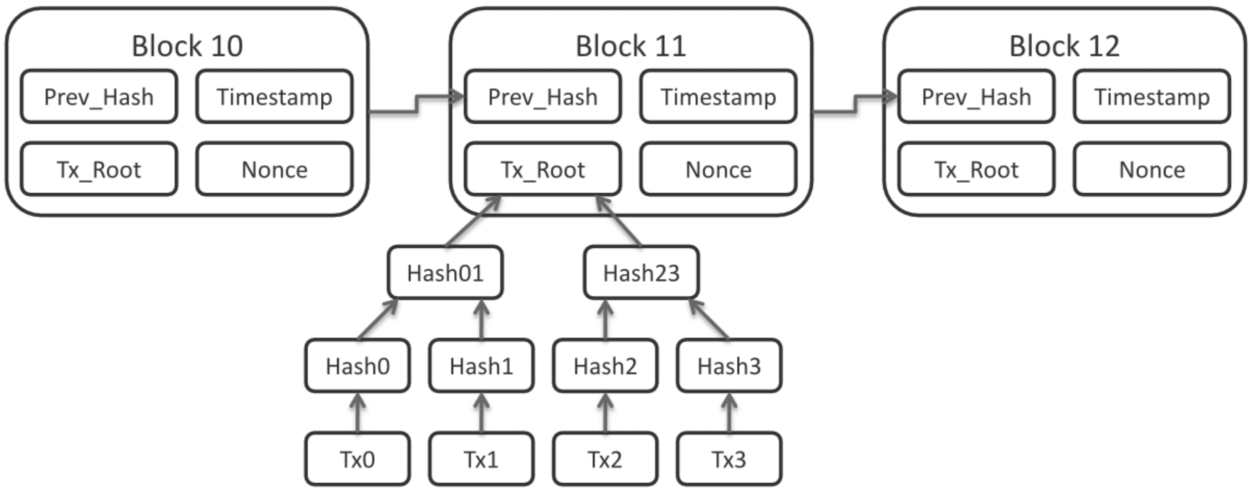


Figure-1: Blockchain Implementation

Part-I: Two-Factor Authentication with Blockchain

Two-factor authentication (2FA) provides an added layer to the existing credential-based system protection as a solution to this drastically growing problem.

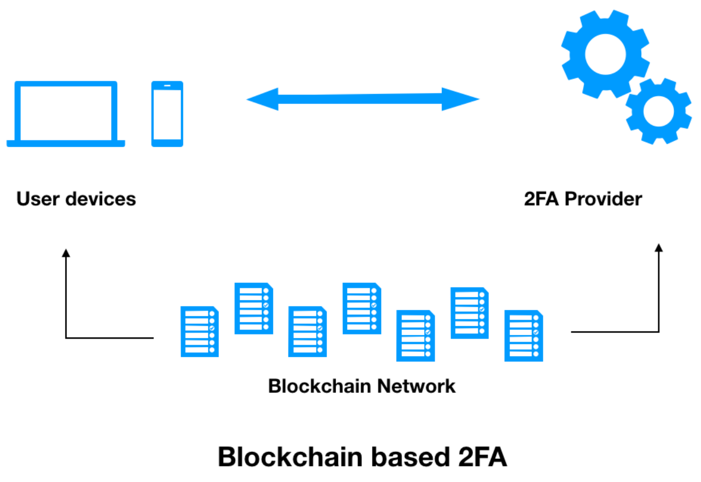


Figure-2: Blockchain-Based 2FA

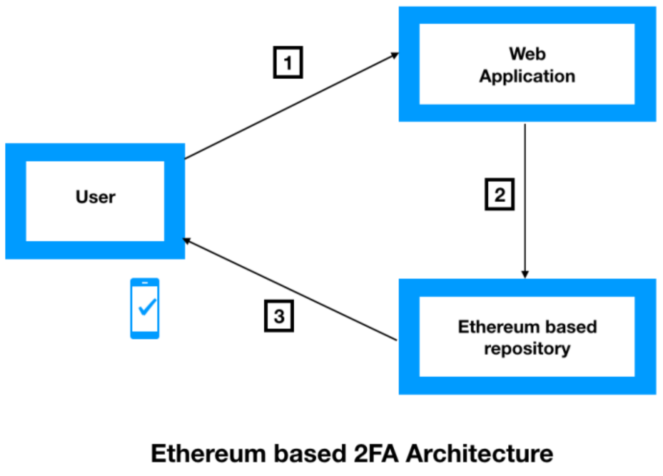


Figure-3:Solution Architecture

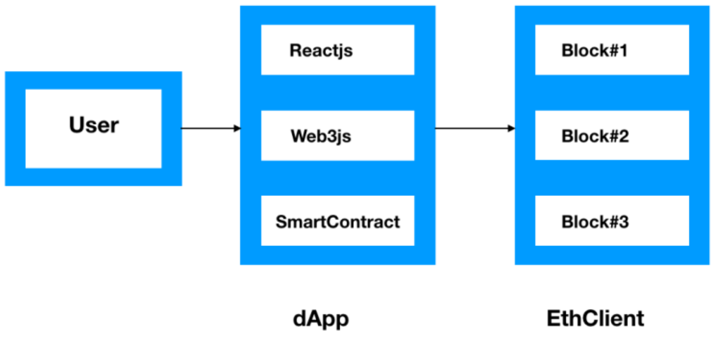


Figure-4: Ethereum based 2FA

Procedure:

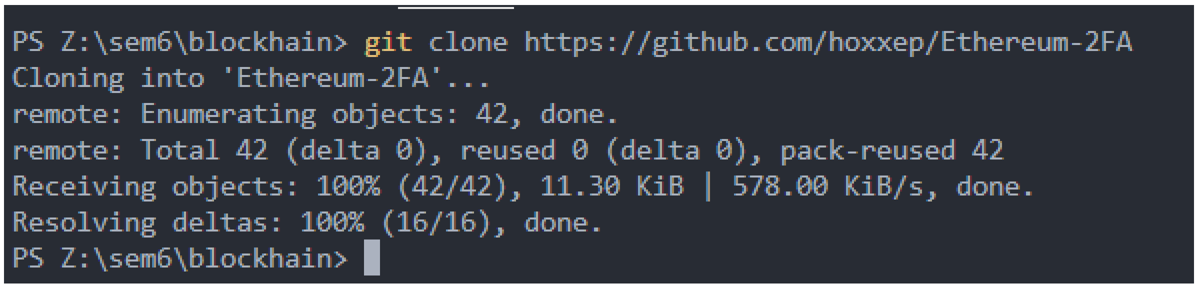
[1] Create a directory under BTA

On windows we create directory named Blockchain inside this we will clone the repo and write all our code

```
PS Z:\sem6\blockhain>
```

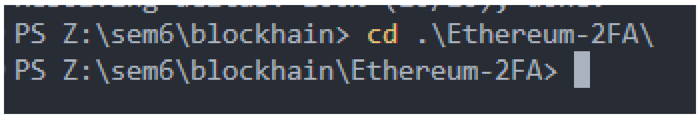
[2] Clone or download the Ethereum-2FA

$ git clone https://github.com/hoxxep/Ethereum-2FA



```
PS Z:\sem6\blockhain> git clone https://github.com/hoxxep/Ethereum-2FA
Cloning into 'Ethereum-2FA'...
remote: Enumerating objects: 42, done.
remote: Total 42 (delta 0), reused 0 (delta 0), pack-reused 42
Receiving objects: 100% (42/42), 11.30 KiB | 578.00 KiB/s, done.
Resolving deltas: 100% (16/16), done.
PS Z:\sem6\blockhain>
```

$cd Ethereum-2FA



```
PS Z:\sem6\blockhain> cd .\Ethereum-2FA\
PS Z:\sem6\blockhain\Ethereum-2FA>
```

The files in the preceding screenshot are explained as follows:

contracts: This folder includes our smart contract, TwoFactorAuth.sol

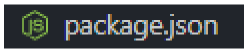
migrations: This folder consists of migration files to deploy the contract to the blockchain

test: This folder consists of server.js, which is responsible for event authentication in our contract

node_modules: This folder includes all the libraries

truffle.js: This configuration file consists of a set of configurations to connect to the blockchain

package.json: This is where we specify a configuration of our projects, such as name and scripts

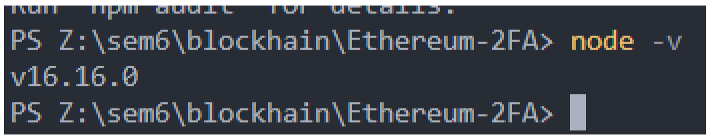


[3] Install Nodejs

I have nodejs pre installed on my laptop

#Check Nodejs version

$node -v

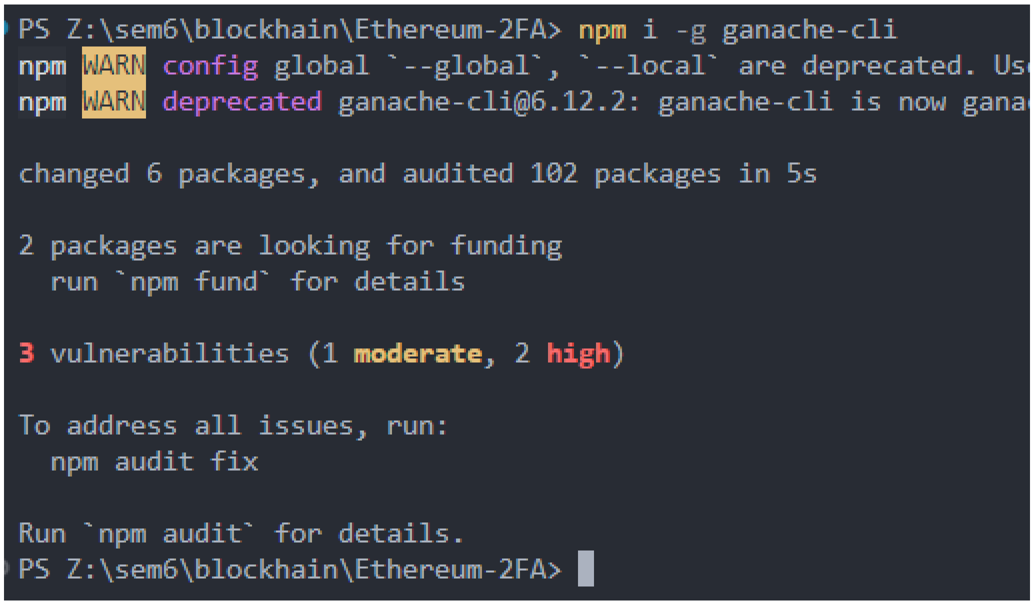


```
PS Z:\sem6\blockhain\Ethereum-2FA> node -v
v16.16.0
PS Z:\sem6\blockhain\Ethereum-2FA>
```

#Turning up Ethereum

#Install ganache-cli

$npm install -g ganache-cli


```
PS Z:\sem6\blockhain\Ethereum-2FA> npm i -g ganache-cli
npm WARN config global `--global`, `--local` are deprecated. Us
npm WARN deprecated ganache-cli@6.12.2: ganache-cli is now gana

changed 6 packages, and audited 102 packages in 5s

2 packages are looking for funding
  run `npm fund` for details

3 vulnerabilities (1 moderate, 2 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS Z:\sem6\blockhain\Ethereum-2FA>
```

Components

The following are the three core components of this project, shown in the following diagram:

- A blockchain network (which we will develop with the Ganache CLI)
- A smart contract
- A server communicating with the blockchain

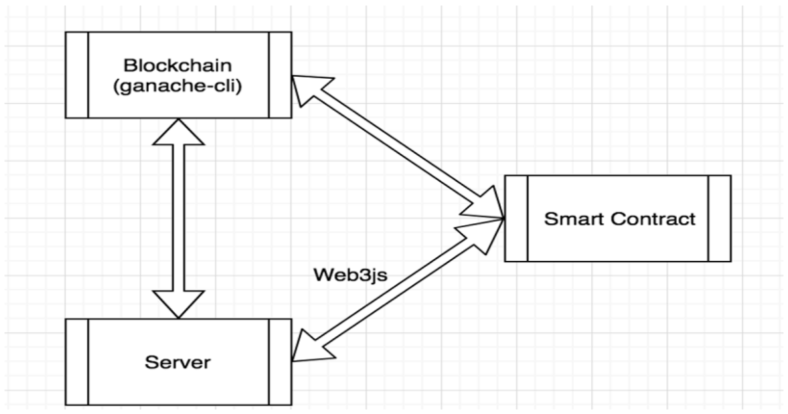
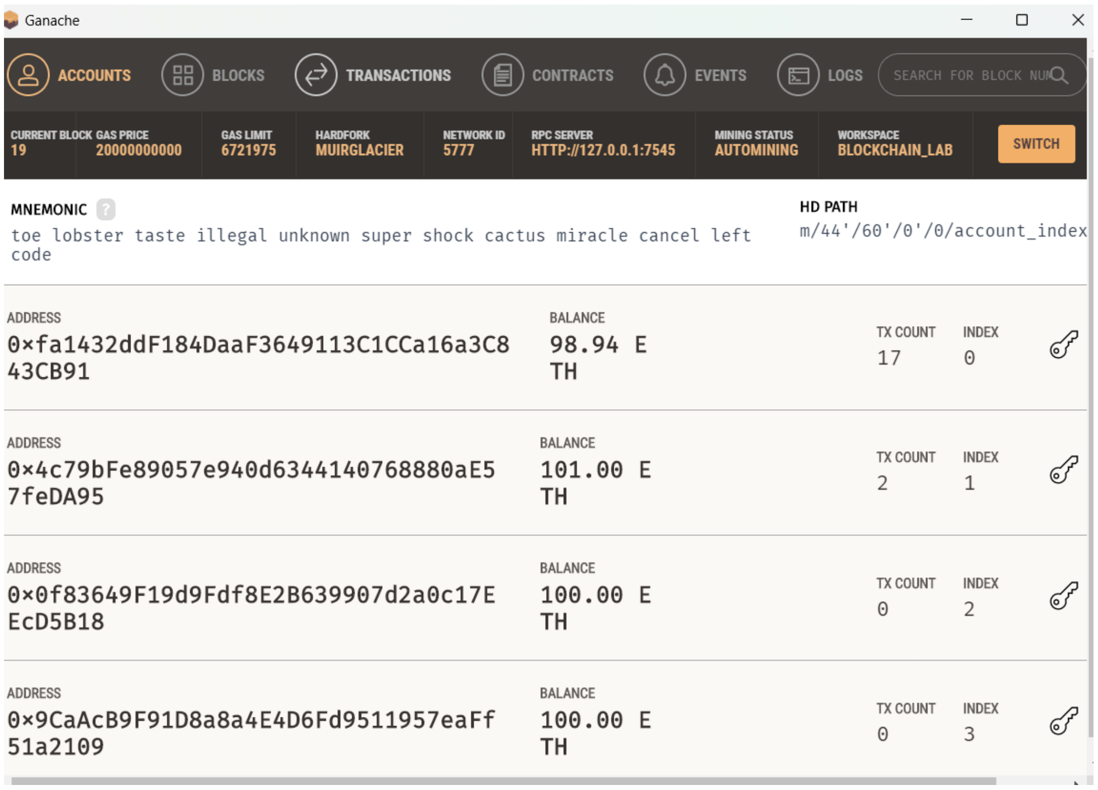


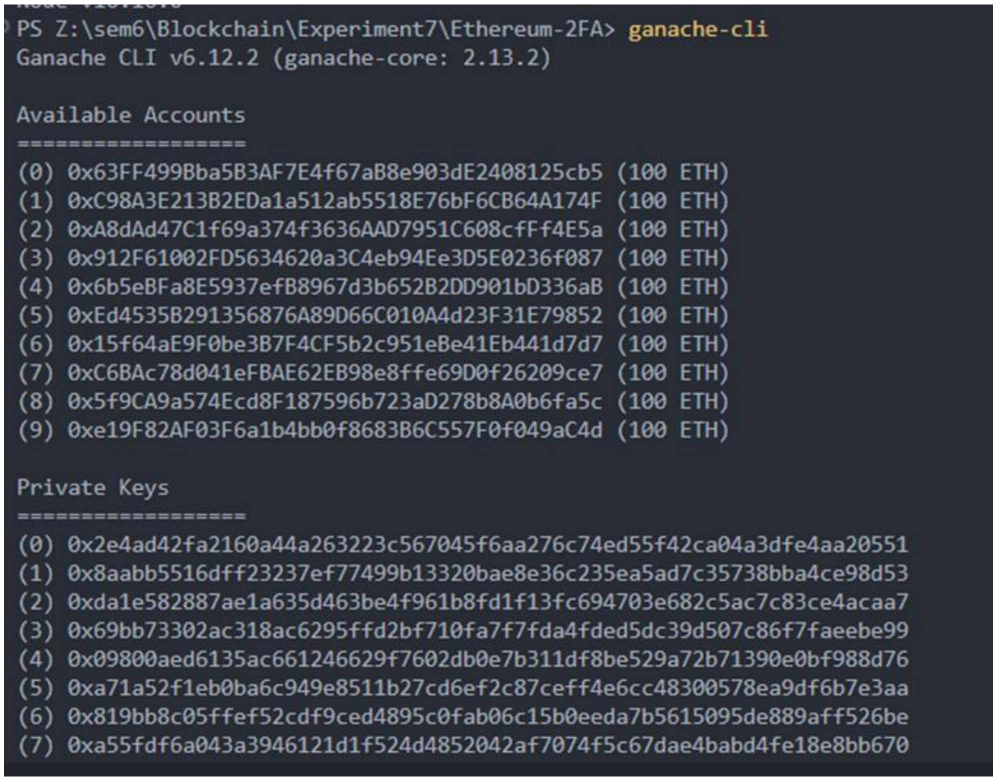
Figure-5: Components of Blockchain-Based 2FA

$ganache-cli

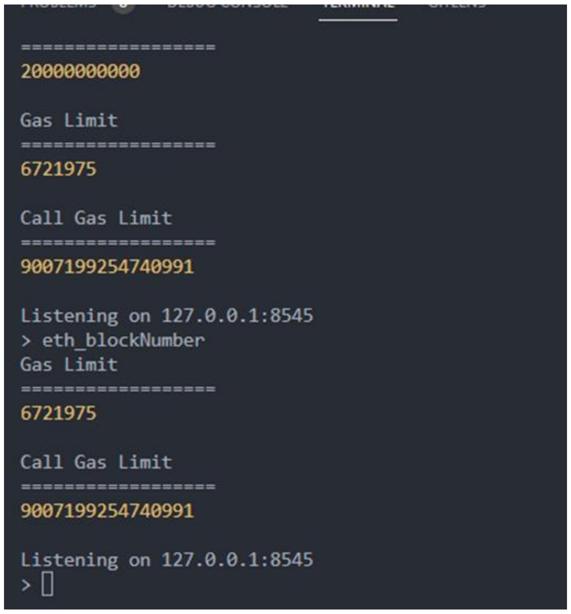


```
CURRENT BLOCK  GAS PRICE    GAS LIMIT  HARDFORK     NETWORK ID  RPC SERVER              MINING STATUS  WORKSPACE
19             20000000000  6721975    MUIRGLACIER  5777        HTTP://127.0.0.1:7545   AUTOMINING     BLOCKCHAIN_LAB

MNEMONIC                                                                       HD PATH
toe lobster taste illegal unknown super shock cactus miracle cancel left      m/44'/60'/0'/0/account_index
code

ADDRESS                                        BALANCE       TX COUNT   INDEX
0x fa1432ddF184DaaF3649113C1CCa16a3C8          98.94 E       17         0
43CB91                                         TH

ADDRESS                                        BALANCE       TX COUNT   INDEX
0x4c79bFe89057e940d6344140768880aE5            101.00 E      2          1
7feDA95                                        TH

ADDRESS                                        BALANCE       TX COUNT   INDEX
0x0f83649F19d9Fdf8E2B639907d2a0c17E            100.00 E      0          2
EcD5B18                                        TH

ADDRESS                                        BALANCE       TX COUNT   INDEX
0x9CaAcB9F91D8a8a4E4D6Fd9511957eaFf            100.00 E      0          3
51a2109                                        TH
```

Start the ganache-cli to listen on port 8545



```
PS Z:\sem6\Blockchain\Experiment7\Ethereum-2FA> ganache-cli
Ganache CLI v6.12.2 (ganache-core: 2.13.2)

Available Accounts
==================
(0) 0x63FF499Bba5B3AF7E4f67aB8e903dE2408125cb5 (100 ETH)
(1) 0xC98A3E213B2EDa1a512ab5518E76bF6CB64A174F (100 ETH)
(2) 0xA8dAd47C1f69a374f3636AAD7951C608cfFf4E5a (100 ETH)
(3) 0x912F61002FD5634620a3C4eb94Ee3D5E0236f087 (100 ETH)
(4) 0x6b5eBFa8E5937efB8967d3b652B2DD901bD336aB (100 ETH)
(5) 0xEd4535B291356876A89D66C010A4d23F31E79852 (100 ETH)
(6) 0x15f64aE9F0be3B7F4CF5b2c951eBe41Eb441d7d7 (100 ETH)
(7) 0xC6BAc78d041eFBAE62EB98e8ffe69D0f26209ce7 (100 ETH)
(8) 0x5f9CA9a574Ecd8F187596b723aD278b8A0b6fa5c (100 ETH)
(9) 0xe19F82AF03F6a1b4bb0f8683B6C557F0f049aC4d (100 ETH)

Private Keys
==================
(0) 0x2e4ad42fa2160a44a263223c567045f6aa276c74ed55f42ca04a3dfe4aa20551
(1) 0x8aabb5516dff23237ef77499b13320bae8e36c235ea5ad7c35738bba4ce98d53
(2) 0xda1e582887ae1a635d463be4f961b8fd1f13fc694703e682c5ac7c83ce4acaa7
(3) 0x69bb73302ac318ac6295ffd2bf710fa7f7fda4fded5dc39d507c86f7faeebe99
(4) 0x09800aed6135ac661246629f7602db0e7b311df8be529a72b71390e0bf988d76
(5) 0xa71a52f1eb0ba6c949e8511b27cd6ef2c87ceff4e6cc48300578ea9df6b7e3aa
(6) 0x819bb8c05ffef52cdf9ced4895c0fab06c15b0eeda7b5615095de889aff526be
(7) 0xa55fdf6a043a3946121d1f524d4852042af7074f5c67dae4babd4fe18e8bb670
```

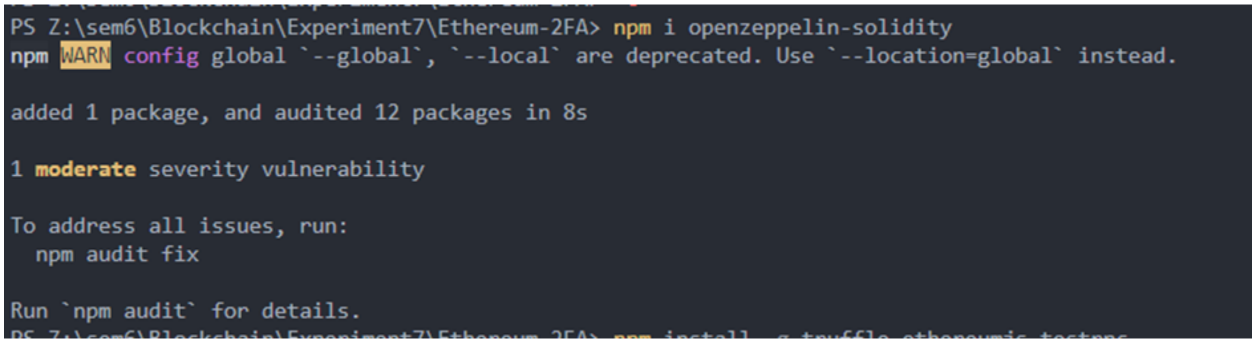


```
==================
20000000000

Gas Limit
==================
6721975

Call Gas Limit
==================
9007199254740991

Listening on 127.0.0.1:8545
> eth_blockNumber
Gas Limit
==================
6721975

Call Gas Limit
==================
9007199254740991

Listening on 127.0.0.1:8545
>
```

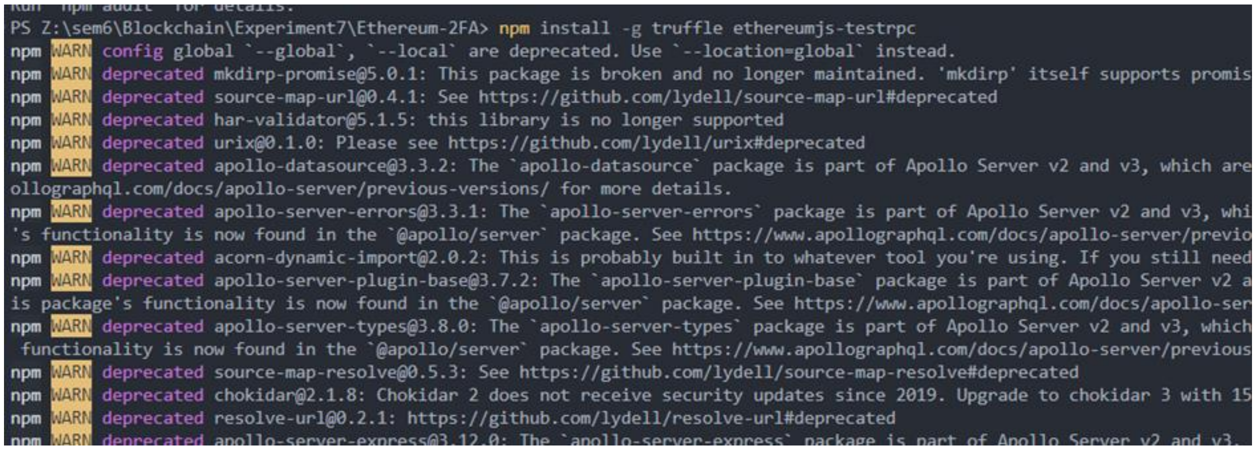
Turning up the smart contract

$cd Ethereum-2FA

Install the necessary dependencies

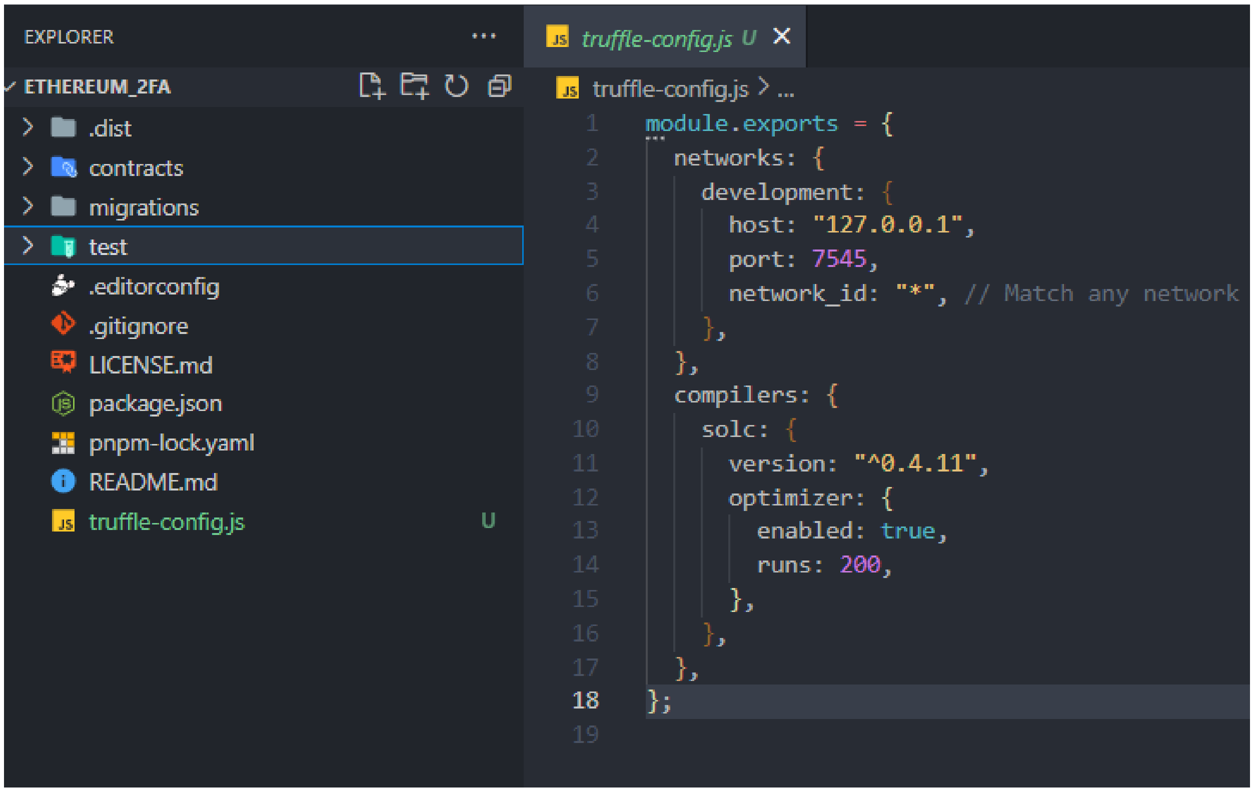


```
PS Z:\sem6\Blockchain\Experiment7\Ethereum-2FA> npm i openzeppelin-solidity
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

added 1 package, and audited 12 packages in 8s

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
```



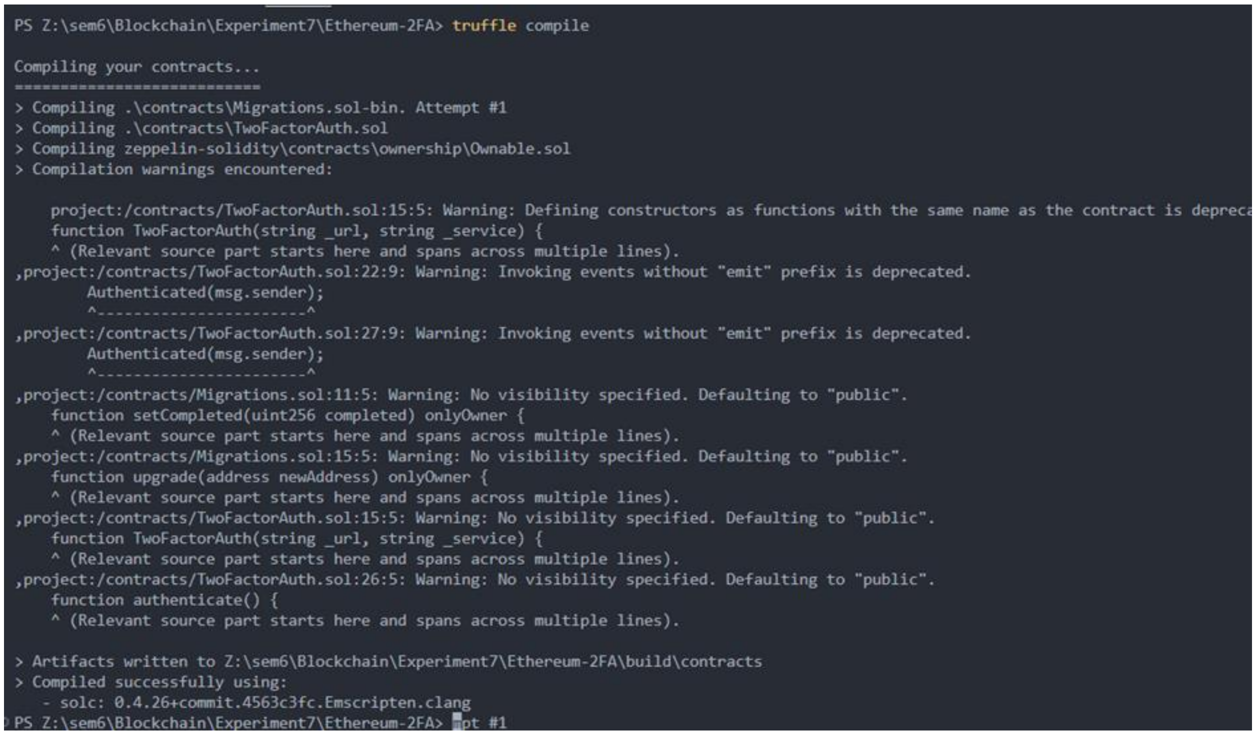
```
PS Z:\sem6\Blockchain\Experiment7\Ethereum-2FA> npm install -g truffle ethereumjs-testrpc
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports promis
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated apollo-datasource@3.3.2: The `apollo-datasource` package is part of Apollo Server v2 and v3, which are
ollographql.com/docs/apollo-server/previous-versions/ for more details.
npm WARN deprecated apollo-server-errors@3.3.1: The `apollo-server-errors` package is part of Apollo Server v2 and v3, whi
's functionality is now found in the `@apollo/server` package. See https://www.apollographql.com/docs/apollo-server/previo
npm WARN deprecated acorn-dynamic-import@2.0.2: This is probably built in to whatever tool you're using. If you still need
npm WARN deprecated apollo-server-plugin-base@3.7.2: The `apollo-server-plugin-base` package is part of Apollo Server v2 a
is package's functionality is now found in the `@apollo/server` package. See https://www.apollographql.com/docs/apollo-ser
npm WARN deprecated apollo-server-types@3.8.0: The `apollo-server-types` package is part of Apollo Server v2 and v3, which
 functionality is now found in the `@apollo/server` package. See https://www.apollographql.com/docs/apollo-server/previous
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated chokidar@2.1.8: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
```

Update truffle.js to truffle-config.js



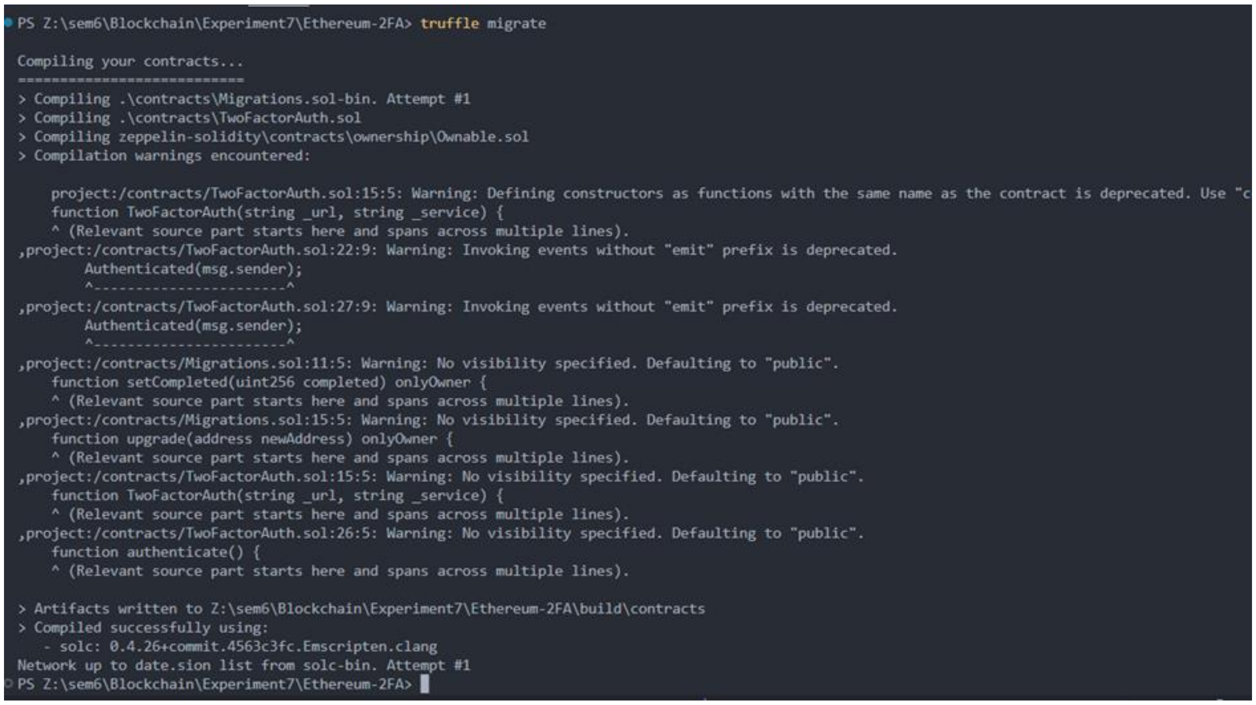
```javascript
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "*", // Match any network
    },
  },
  compilers: {
    solc: {
      version: "^0.4.11",
      optimizer: {
        enabled: true,
        runs: 200,
      },
    },
  },
};
```

Run the truffle compile command



```
PS Z:\sem6\Blockchain\Experiment7\Ethereum-2FA> truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol-bin. Attempt #1
> Compiling .\contracts\TwoFactorAuth.sol
> Compiling zeppelin-solidity\contracts\ownership\Ownable.sol
> Compilation warnings encountered:

    project:/contracts/TwoFactorAuth.sol:15:5: Warning: Defining constructors as functions with the same name as the contract is depreca
    function TwoFactorAuth(string _url, string _service) {
    ^ (Relevant source part starts here and spans across multiple lines).
,project:/contracts/TwoFactorAuth.sol:22:9: Warning: Invoking events without "emit" prefix is deprecated.
        Authenticated(msg.sender);
        ^-----------------------^
,project:/contracts/TwoFactorAuth.sol:27:9: Warning: Invoking events without "emit" prefix is deprecated.
        Authenticated(msg.sender);
        ^-----------------------^
,project:/contracts/Migrations.sol:11:5: Warning: No visibility specified. Defaulting to "public".
    function setCompleted(uint256 completed) onlyOwner {
    ^ (Relevant source part starts here and spans across multiple lines).
,project:/contracts/Migrations.sol:15:5: Warning: No visibility specified. Defaulting to "public".
    function upgrade(address newAddress) onlyOwner {
    ^ (Relevant source part starts here and spans across multiple lines).
,project:/contracts/TwoFactorAuth.sol:15:5: Warning: No visibility specified. Defaulting to "public".
    function TwoFactorAuth(string _url, string _service) {
    ^ (Relevant source part starts here and spans across multiple lines).
,project:/contracts/TwoFactorAuth.sol:26:5: Warning: No visibility specified. Defaulting to "public".
    function authenticate() {
    ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to Z:\sem6\Blockchain\Experiment7\Ethereum-2FA\build\contracts
> Compiled successfully using:
   - solc: 0.4.26+commit.4563c3fc.Emscripten.clang
PS Z:\sem6\Blockchain\Experiment7\Ethereum-2FA> mpt #1
```

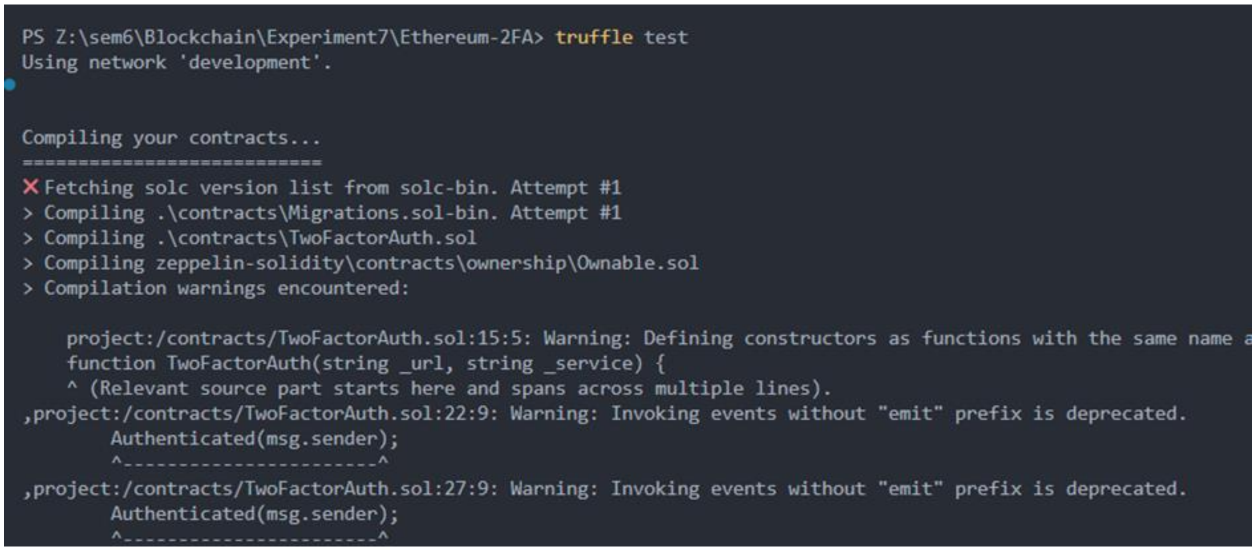
Run truffle migrate command



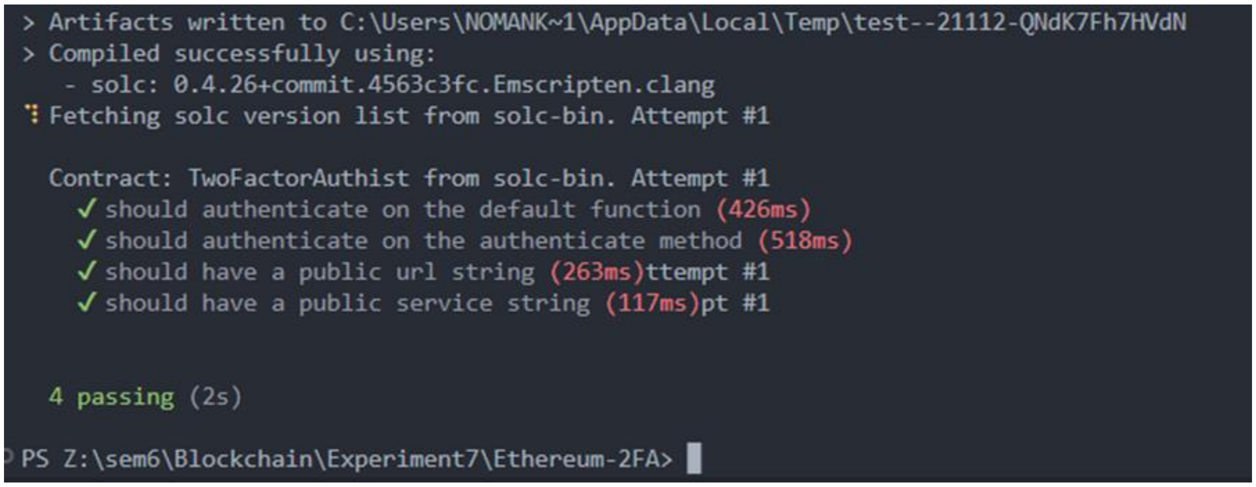
```
PS Z:\sem6\Blockchain\Experiment7\Ethereum-2FA> truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol-bin. Attempt #1
> Compiling .\contracts\TwoFactorAuth.sol
> Compiling zeppelin-solidity\contracts\ownership\Ownable.sol
> Compilation warnings encountered:

    project:/contracts/TwoFactorAuth.sol:15:5: Warning: Defining constructors as functions with the same name as the contract is deprecated. Use "c
    function TwoFactorAuth(string _url, string _service) {
    ^ (Relevant source part starts here and spans across multiple lines).
,project:/contracts/TwoFactorAuth.sol:22:9: Warning: Invoking events without "emit" prefix is deprecated.
        Authenticated(msg.sender);
        ^-----------------------^
,project:/contracts/TwoFactorAuth.sol:27:9: Warning: Invoking events without "emit" prefix is deprecated.
        Authenticated(msg.sender);
        ^-----------------------^
,project:/contracts/Migrations.sol:11:5: Warning: No visibility specified. Defaulting to "public".
    function setCompleted(uint256 completed) onlyOwner {
    ^ (Relevant source part starts here and spans across multiple lines).
,project:/contracts/Migrations.sol:15:5: Warning: No visibility specified. Defaulting to "public".
    function upgrade(address newAddress) onlyOwner {
    ^ (Relevant source part starts here and spans across multiple lines).
,project:/contracts/TwoFactorAuth.sol:15:5: Warning: No visibility specified. Defaulting to "public".
    function TwoFactorAuth(string _url, string _service) {
    ^ (Relevant source part starts here and spans across multiple lines).
,project:/contracts/TwoFactorAuth.sol:26:5: Warning: No visibility specified. Defaulting to "public".
    function authenticate() {
    ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to Z:\sem6\Blockchain\Experiment7\Ethereum-2FA\build\contracts
> Compiled successfully using:
   - solc: 0.4.26+commit.4563c3fc.Emscripten.clang
Network up to date.sion list from solc-bin. Attempt #1
PS Z:\sem6\Blockchain\Experiment7\Ethereum-2FA> █
```

$truffle test

Now let's test our contracts using the truffle test command



```
PS Z:\sem6\Blockchain\Experiment7\Ethereum-2FA> truffle test
Using network 'development'.


Compiling your contracts...
===========================
✕ Fetching solc version list from solc-bin. Attempt #1
> Compiling .\contracts\Migrations.sol-bin. Attempt #1
> Compiling .\contracts\TwoFactorAuth.sol
> Compiling zeppelin-solidity\contracts\ownership\Ownable.sol
> Compilation warnings encountered:

    project:/contracts/TwoFactorAuth.sol:15:5: Warning: Defining constructors as functions with the same name a
    function TwoFactorAuth(string _url, string _service) {
    ^ (Relevant source part starts here and spans across multiple lines).
,project:/contracts/TwoFactorAuth.sol:22:9: Warning: Invoking events without "emit" prefix is deprecated.
        Authenticated(msg.sender);
        ^-----------------------^
,project:/contracts/TwoFactorAuth.sol:27:9: Warning: Invoking events without "emit" prefix is deprecated.
        Authenticated(msg.sender);
        ^-----------------------^
```

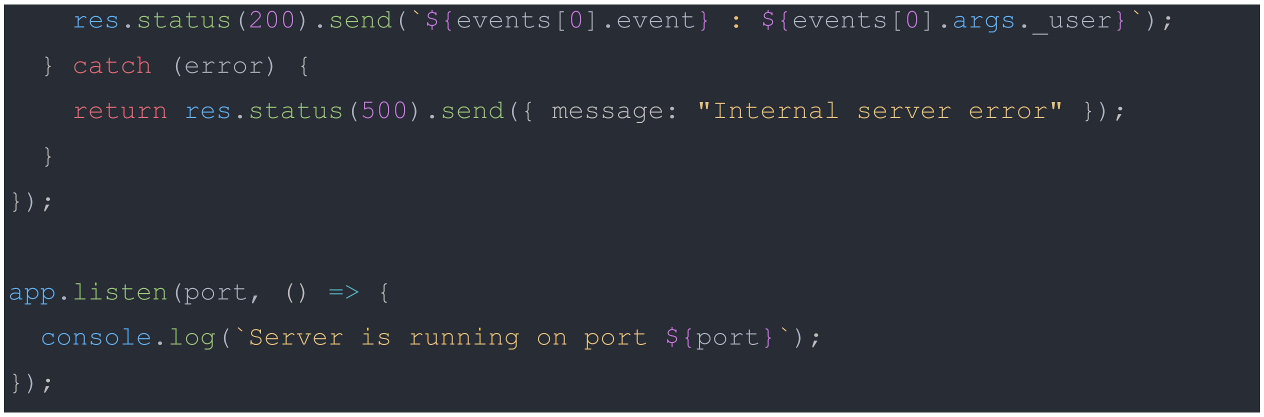


```
> Artifacts written to C:\Users\NOMANK~1\AppData\Local\Temp\test--21112-QNdK7Fh7HVdN
> Compiled successfully using:
   - solc: 0.4.26+commit.4563c3fc.Emscripten.clang
⠹ Fetching solc version list from solc-bin. Attempt #1

  Contract: TwoFactorAuthist from solc-bin. Attempt #1
    ✓ should authenticate on the default function (426ms)
    ✓ should authenticate on the authenticate method (518ms)
    ✓ should have a public url string (263ms)ttempt #1
    ✓ should have a public service string (117ms)pt #1


  4 passing (2s)

PS Z:\sem6\Blockchain\Experiment7\Ethereum-2FA> ▌
```

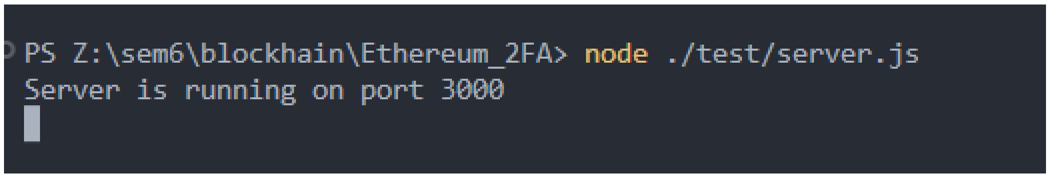
Figure-8:Ethereum-2FA

Server Code in express js with web3

```
require("./helpers/assert");
const express = require("express");
const app = express();
const port = 3000;
const bodyParser = require("body-parser");
const Web3 = require("web3");
const web3 = new Web3("http://127.0.0.1:7545");
const contract = require("truffle-contract");
const twoFactorAuth = contract(
  require("../build/contracts/TwoFactorAuth.json")
);
twoFactorAuth.setProvider(web3.currentProvider);
const twoFactorAuthInstance = twoFactorAuth.deployed();

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.get("/", async (req, res) => {
  try {
    const instance = await twoFactorAuth.deployed();
    const accounts = await web3.eth.getAccounts();
    const result = await instance.authenticate({ from: accounts[1] });
    const events = result.logs;
```

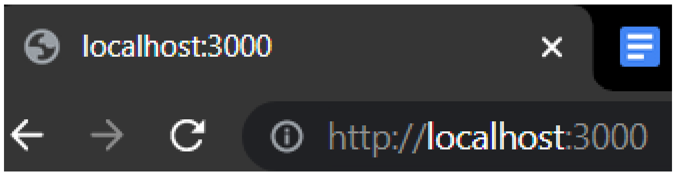


```
    res.status(200).send(`${events[0].event} : ${events[0].args._user}`);
  } catch (error) {
    return res.status(500).send({ message: "Internal server error" });
  }
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```



```
PS Z:\sem6\blockhain\Ethereum_2FA> node ./test/server.js
Server is running on port 3000
```

# Open Google Chrome and access the localhost on port 3000



Successful authentication:

Authenticated : 0x4c79bFe89057e940d6344140768880aE57feDA95

Conclusion: In this experiment,

1. Created a private Ethereum blockchain by configuring network parameters and utilizing Ethereum clients.
2. Developed a 2FA system using Solidity language and a smart contract, which was deployed and executed through the Ethereum network.
3. The Ethereum Blockchain can compile and run this security solution, providing robust and reliable security.
4. Technical knowledge and expertise are required to implement these procedures effectively.

References:

[1] Two-factor authentication through an Ethereum contract.

https://github.com/hoxxep/Ethereum-2FA

[2] How To Install Node.js on Ubuntu 20.04

How To Install Node.js on Ubuntu 20.04 | DigitalOcean