Configuration Manual

MSc Research Project
Msc Cybersecurity

Vasu Singh
Student ID: 22243674

School of Computing
National College of Ireland

Supervisor: Prof. Raza Ul Mustafa

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vasu Singh
Student ID:	22243674
Programme:	Msc Cybersecurity
Year:	2023-24
Module:	Msc Reserach Project
Supervisor:	Prof. Raza Ul Mustafa
Submission Due Date:	12/10/2024
Project Title:	Blockchain-Integrated Identity and Access Management Framework for Secure IoT Device Management
Word Count:	558
Page Count:	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vasu Singh
Date:	16th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vasu Singh
22243674

1 Introduction

To set up and configure the Blockchain-Integrated Identity and Access Management Framework for Secure IoT Device Management, the detailed instructions provided in this configuration manual. System prerequisites, software tool installation instructions, and the process of integrating smart contracts with the Ethereum blockchain and Keycloak for identity management are all described.

2 System Configuration

The below table provide details about the System Configuration used to implement the model.

Component	**Specification**
Model Name	MacBook Air
Model ID	MacBookAir 10,1
Processor/Chip	Apple M1 2020
RAM/Memory	8 GB
Total Number of Cores	8 (4 performance and 4 efficiency)
Storage	50 GB free space
Operating System	macOS Monterey
Operating System Version	12.2.1

Table 1: System Configuration

3 Software Configuration

This section discuss about the software and tools used to carry out the research. The below table the all the details.

Software	Version	Dependencies
Node.js	v14.17.0 or later	NPM
NPM	v6.14.13 (comes with Node.js)	None
Truffle	v5.4.29	Node.js, NPM
Ganache	v2.5.4	Node.js, NPM
OpenSSL	v1.1.1 or later	None
Keycloak	v16.1.1	JDK 8+
Solidity	v0.8.13	Node.js, NPM
Slither	v0.8.3	Python 3.x, Pip
Web3.js	v1.5.2	Node.js, NPM
LaTeX (For reporting)	TeX 2021 or later	None
Google Chrome	127.0.6533.26	None

Table 2: Software Configuration for IAM Framework Setup

4 Installation of Tools and Software

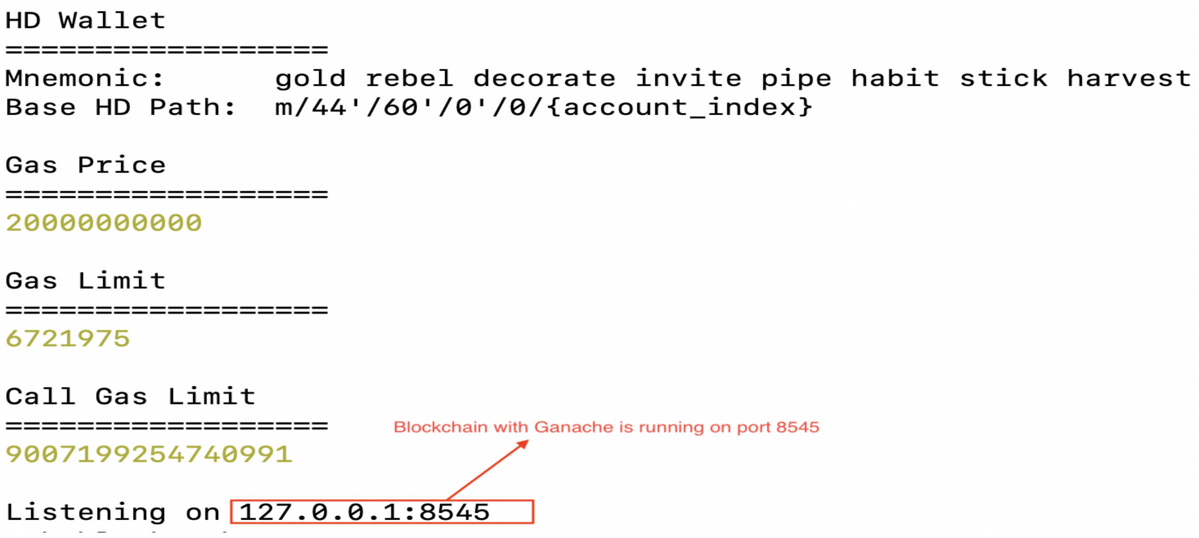
4.1 Installation of Node.js and NPM

Below command used to install node packages whihc are required for interacting with Blockchain.

brew install node[1]

[1] https://formulae.brew.sh/formula/node

4.2 Truffle and Ganache



```
HD Wallet
==================
Mnemonic:      gold rebel decorate invite pipe habit stick harvest
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
==================
20000000000

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

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

Listening on 127.0.0.1:8545
```

Blockchain with Ganache is running on port 8545

Figure 1: Model Blockchain IP

This is required to setup local test environment for Ethereum blockchain.

npm install -g truffle [2]

npm install -g ganache-cli [3]

To start Ganache, run below command.

ganache-cli [4]

4.3 Keycloak Setup

Manually downloaded the Keycloak[5] package in installed on the system.

After downloading the .zip file, unzipped it and ran below command to start the keycloak server over port 8080.

./bin/kc.sh -Djboss.socket.binding.port-offset=100 [6]

And we can see from below figure that keycloak is accessible on '**http://localhost:8080/**'

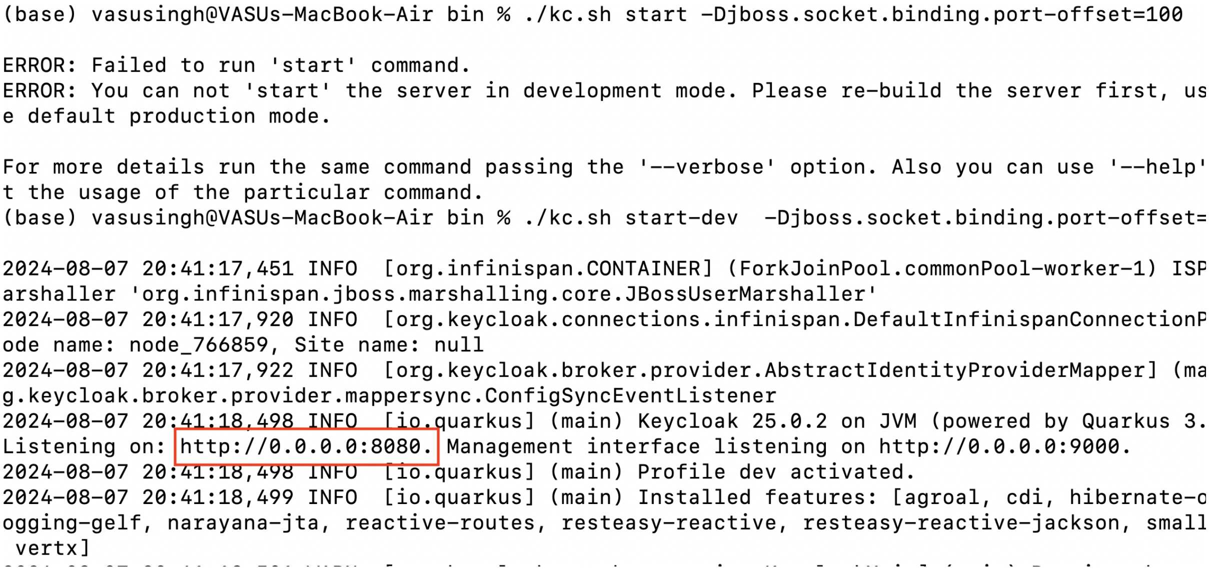
[2] https://archive.trufflesuite.com/docs/truffle/how-to/install/

[3] https://www.npmjs.com/package/ganache

[4] https://www.npmjs.com/package/ganache

[5] https://www.keycloak.org/downloads

[6] https://www.keycloak.org/server/configuration



```
(base) vasusingh@VASUs-MacBook-Air bin % ./kc.sh start -Djboss.socket.binding.port-offset=100

ERROR: Failed to run 'start' command.
ERROR: You can not 'start' the server in development mode. Please re-build the server first, us
e default production mode.

For more details run the same command passing the '--verbose' option. Also you can use '--help'
t the usage of the particular command.
(base) vasusingh@VASUs-MacBook-Air bin % ./kc.sh start-dev  -Djboss.socket.binding.port-offset=

2024-08-07 20:41:17,451 INFO  [org.infinispan.CONTAINER] (ForkJoinPool.commonPool-worker-1) ISP
arshaller 'org.infinispan.jboss.marshalling.core.JBossUserMarshaller'
2024-08-07 20:41:17,920 INFO  [org.keycloak.connections.infinispan.DefaultInfinispanConnectionP
ode name: node_766859, Site name: null
2024-08-07 20:41:17,922 INFO  [org.keycloak.broker.provider.AbstractIdentityProviderMapper] (ma
g.keycloak.broker.provider.mappersync.ConfigSyncEventListener
2024-08-07 20:41:18,498 INFO  [io.quarkus] (main) Keycloak 25.0.2 on JVM (powered by Quarkus 3.
Listening on: http://0.0.0.0:8080. Management interface listening on http://0.0.0.0:9000.
2024-08-07 20:41:18,498 INFO  [io.quarkus] (main) Profile dev activated.
2024-08-07 20:41:18,499 INFO  [io.quarkus] (main) Installed features: [agroal, cdi, hibernate-o
ogging-gelf, narayana-jta, reactive-routes, resteasy-reactive, resteasy-reactive-jackson, small
 vertx]
```

Figure 2: Keycloak IP

Realm creation: Created '**IoTrealm**' by adding new over admin console.

Client Creation: created new client named '**Blockchain-client**' to obtain client ID and secret which further used to obtain JWT Tokens.

User creation: created new user and assign him roles, user credentials are also required to generate JWT token.

4.4 OpenSSL installation

OpenSSL is used for generation and management of PKI certificates.

brew install openssl

5 Solidity smart contract

Smart contract solidity code IoT device management.

// SPDX-License-Identifier: MIT pragma solidity = 0.8.13;

contract IoTDevice struct Device string id; string owner; string deviceType; string publicKey; string certificate;

mapping(string =¿ Device) public devices;

function createDevice(string memory id, string memory owner, string memory deviceType, string memory publicKey, string memory certificate) public devices[id] = Device(id, owner, deviceType, publicKey, certificate);

function getDevice(string memory id) public view returns (string memory, string memory, string memory, string memory, string memory) Device memory device = devices[id]; return (device.id, device.owner, device.deviceType, device.publicKey, device.certificate);

Migrations scripts written in JSON to compile and deploy smart contracts.

Below command used to deploy the smart contracts.

Migration.js

const Migrations = artifacts.require("Migrations");

module.exports = function (deployer) deployer.deploy(Migrations); ;

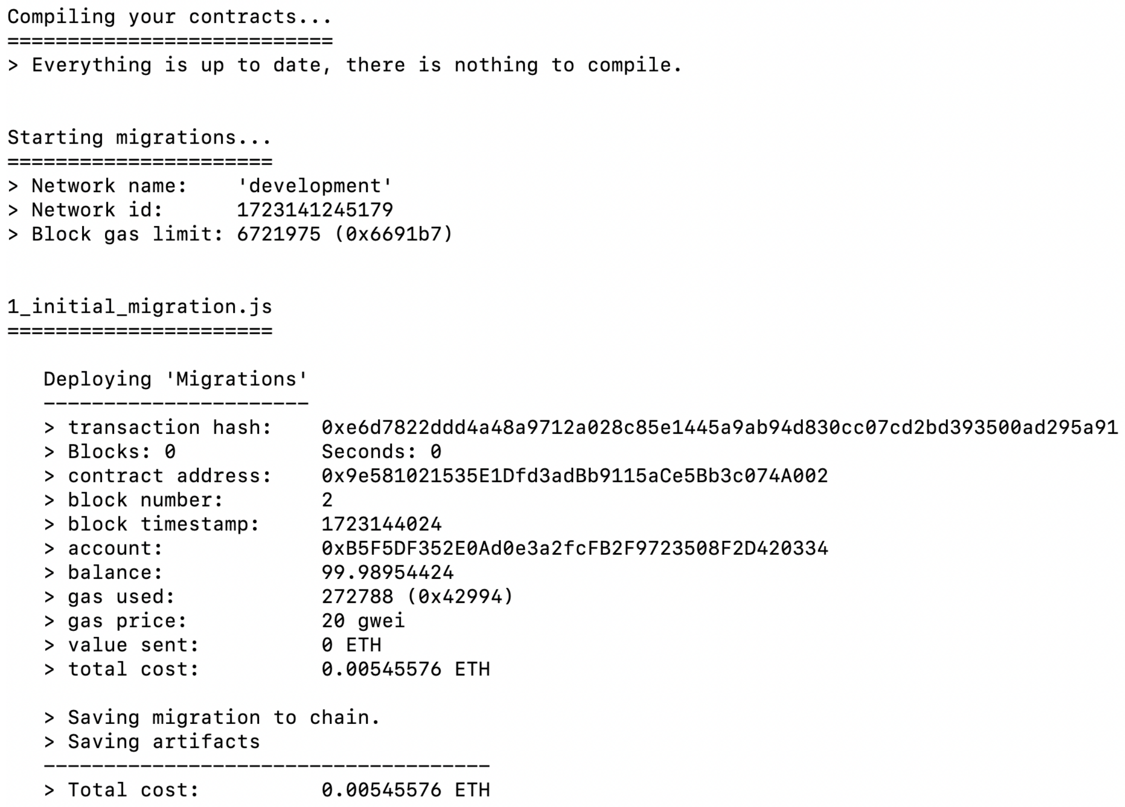
Deployment script

const Migrations = artifacts.require(”Migrations”);

module.exports = function (deployer) deployer.deploy(Migrations); ;

truffle compile

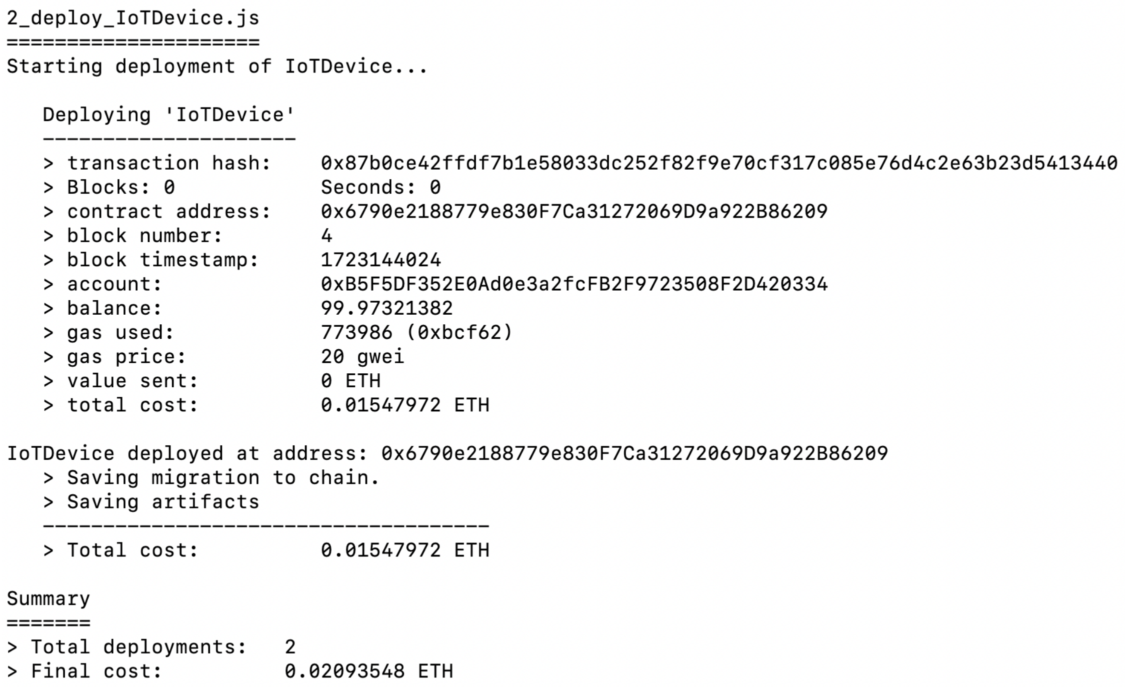
truffle migrate –network development [7]



```
Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.


Starting migrations...
======================
> Network name:    'development'
> Network id:      1723141245179
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0xe6d7822ddd4a48a9712a028c85e1445a9ab94d830cc07cd2bd393500ad295a91
   > Blocks: 0            Seconds: 0
   > contract address:    0x9e581021535E1Dfd3adBb9115aCe5Bb3c074A002
   > block number:        2
   > block timestamp:     1723144024
   > account:             0xB5F5DF352E0Ad0e3a2fcFB2F9723508F2D420334
   > balance:             99.98954424
   > gas used:            272788 (0x42994)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00545576 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00545576 ETH
```

Figure 3: Migration script Deployment

[7] https://archive.trufflesuite.com/docs/truffle/concepts/networks-and-app-deployment/

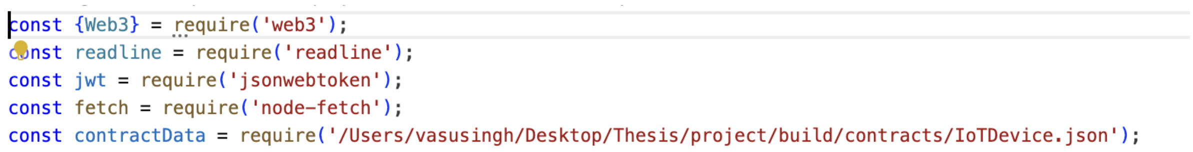


```
2_deploy_IoTDevice.js
=====================
Starting deployment of IoTDevice...

   Deploying 'IoTDevice'
   ---------------------
   > transaction hash:    0x87b0ce42ffdf7b1e58033dc252f82f9e70cf317c085e76d4c2e63b23d5413440
   > Blocks: 0            Seconds: 0
   > contract address:    0x6790e2188779e830F7Ca31272069D9a922B86209
   > block number:        4
   > block timestamp:     1723144024
   > account:             0xB5F5DF352E0Ad0e3a2fcFB2F9723508F2D420334
   > balance:             99.97321382
   > gas used:            773986 (0xbcf62)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.01547972 ETH

IoTDevice deployed at address: 0x6790e2188779e830F7Ca31272069D9a922B86209
   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.01547972 ETH

Summary
=======
> Total deployments:   2
> Final cost:          0.02093548 ETH
```

Figure 4: Device script Deployment

6 Node.js Backend Code to interact with Blockchain

6.1 Importing libraries and modules

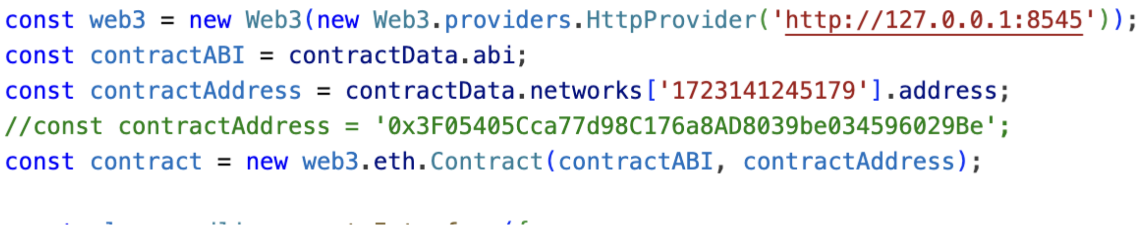


```
const {Web3} = require('web3');
const readline = require('readline');
const jwt = require('jsonwebtoken');
const fetch = require('node-fetch');
const contractData = require('/Users/vasusingh/Desktop/Thesis/project/build/contracts/IoTDevice.json');
```

Figure 5: importing required modules

6.2 Web3 setup and smart contracts

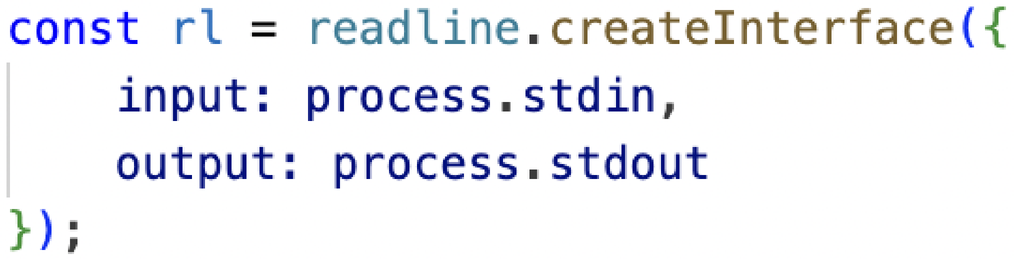
Web3 is setup and connected ethereum blockchain running on '**http://127.0.0.1:8545**' and along with address of smart contract.



```
const web3 = new Web3(new Web3.providers.HttpProvider('http://127.0.0.1:8545'));
const contractABI = contractData.abi;
const contractAddress = contractData.networks['1723141245179'].address;
//const contractAddress = '0x3F05405Cca77d98C176a8AD8039be034596029Be';
const contract = new web3.eth.Contract(contractABI, contractAddress);
```

Figure 6: Web3 setup and smart contracts

6.3 Interface for user input

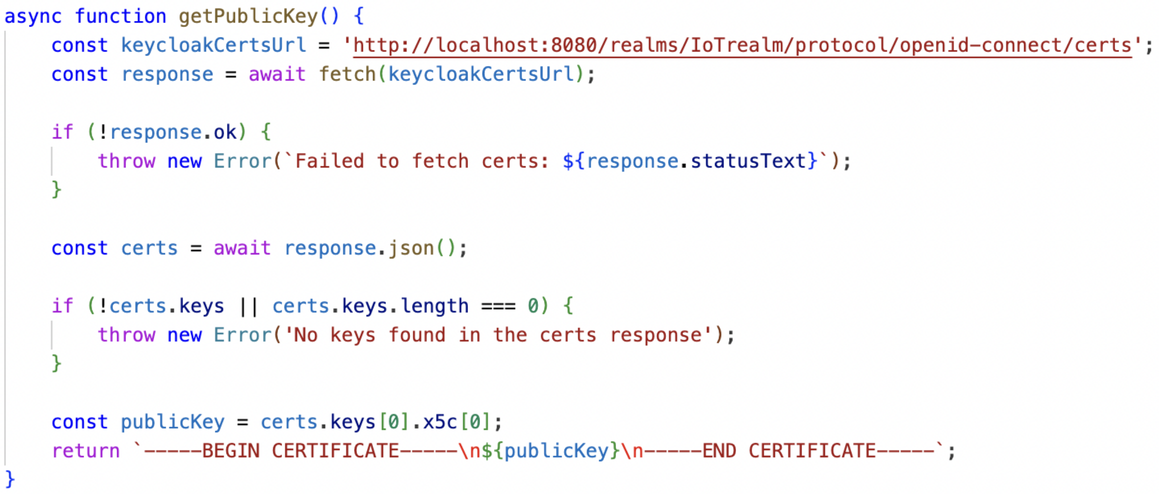


```
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});
```

Figure 7: defined user interface

6.4 Fetching Public Key from Keycloak

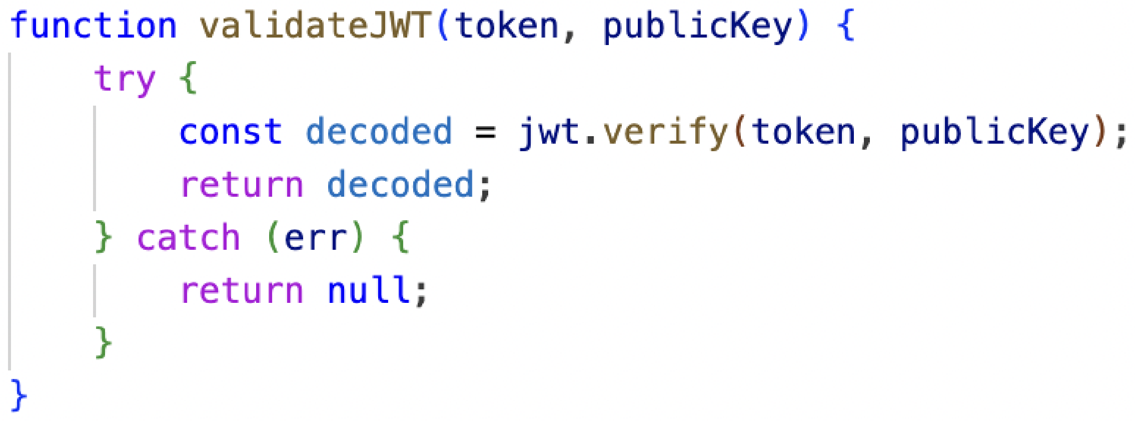
Keycloak certificate endpoint- '**http://localhost:8080/realms/IoTrealm/protocol/openid-connect/certs**'



```
async function getPublicKey() {
    const keycloakCertsUrl = 'http://localhost:8080/realms/IoTrealm/protocol/openid-connect/certs';
    const response = await fetch(keycloakCertsUrl);

    if (!response.ok) {
        throw new Error(`Failed to fetch certs: ${response.statusText}`);
    }

    const certs = await response.json();

    if (!certs.keys || certs.keys.length === 0) {
        throw new Error('No keys found in the certs response');
    }

    const publicKey = certs.keys[0].x5c[0];
    return `-----BEGIN CERTIFICATE-----\n${publicKey}\n-----END CERTIFICATE-----`;
}
```

Figure 8: Public key function

6.5 JWT Validation Function

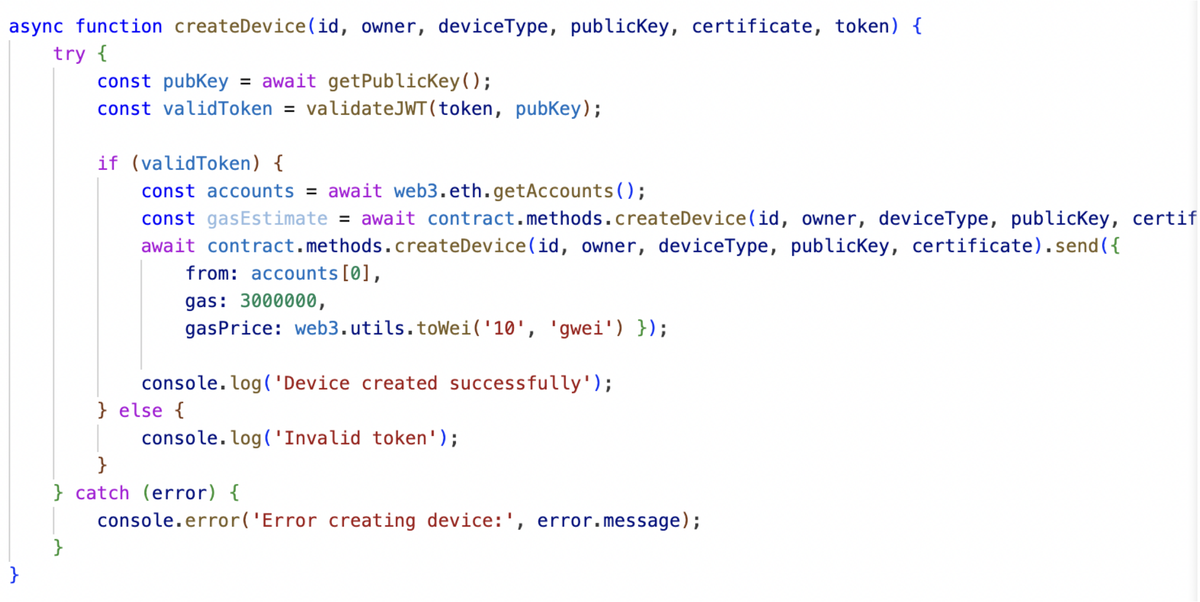


```
function validateJWT(token, publicKey) {
    try {
        const decoded = jwt.verify(token, publicKey);
        return decoded;
    } catch (err) {
        return null;
    }
}
```

Figure 9: Token validation function

6.6 Registering a Device on the Blockchain

Create Command: Prompts the user for device details and a JWT token, then calls createDevice.

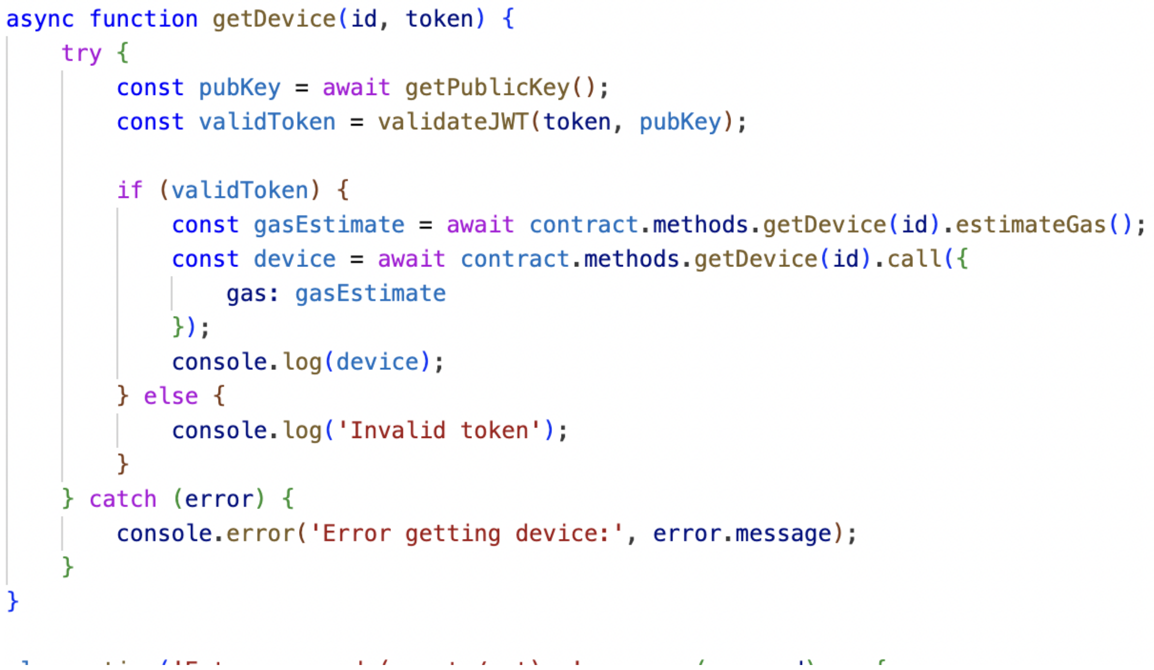


```
async function createDevice(id, owner, deviceType, publicKey, certificate, token) {
    try {
        const pubKey = await getPublicKey();
        const validToken = validateJWT(token, pubKey);

        if (validToken) {
            const accounts = await web3.eth.getAccounts();
            const gasEstimate = await contract.methods.createDevice(id, owner, deviceType, publicKey, certif
            await contract.methods.createDevice(id, owner, deviceType, publicKey, certificate).send({
                from: accounts[0],
                gas: 3000000,
                gasPrice: web3.utils.toWei('10', 'gwei') });

            console.log('Device created successfully');
        } else {
            console.log('Invalid token');
        }
    } catch (error) {
        console.error('Error creating device:', error.message);
    }
}
```

Figure 10: Create device function

6.7 Fetching Device Information

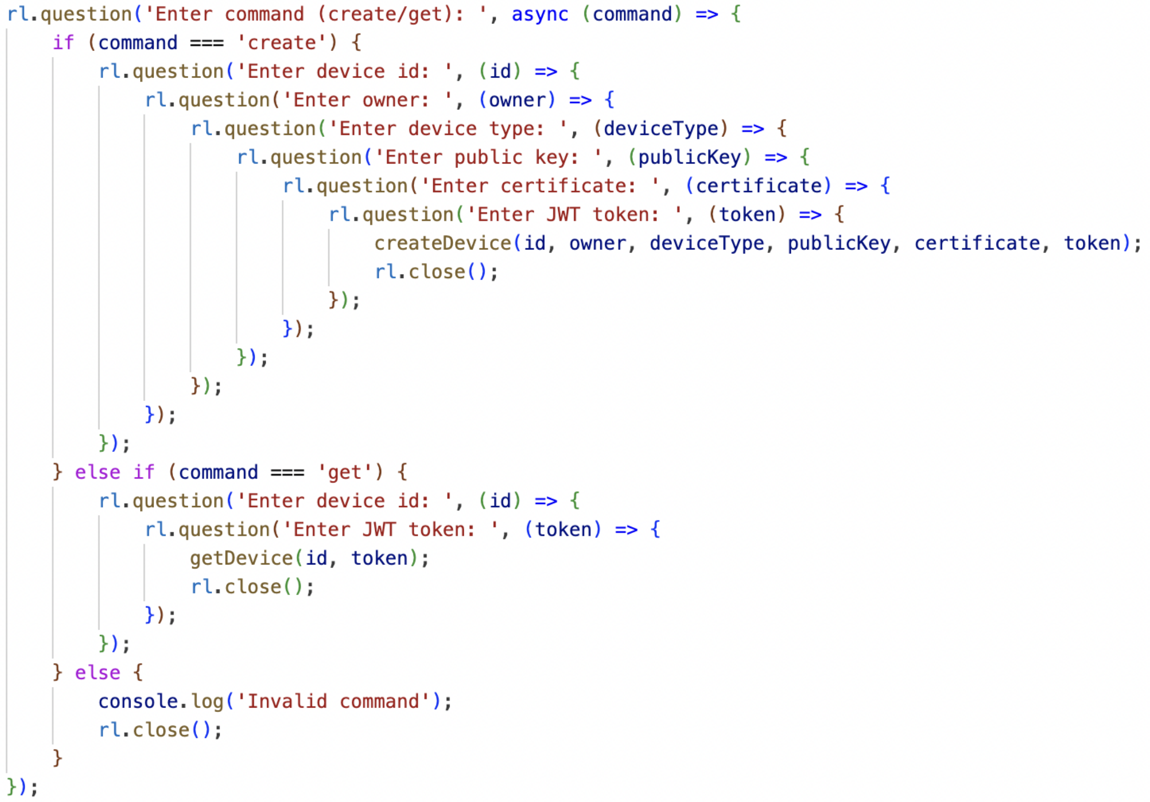
Get Command: Prompts the user for a device ID and a JWT token, then calls getDevice.



```
async function getDevice(id, token) {
    try {
        const pubKey = await getPublicKey();
        const validToken = validateJWT(token, pubKey);

        if (validToken) {
            const gasEstimate = await contract.methods.getDevice(id).estimateGas();
            const device = await contract.methods.getDevice(id).call({
                gas: gasEstimate
            });
            console.log(device);
        } else {
            console.log('Invalid token');
        }
    } catch (error) {
        console.error('Error getting device:', error.message);
    }
}
```

Figure 11: Get device function

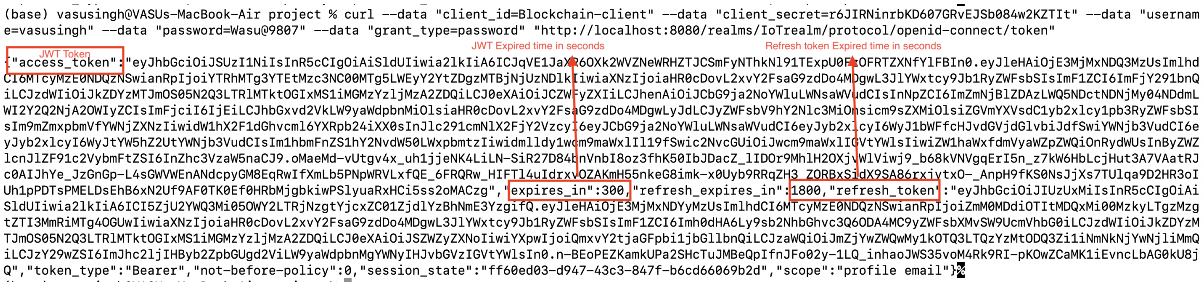
6.8 Command-Line Interaction



```
rl.question('Enter command (create/get): ', async (command) => {
    if (command === 'create') {
        rl.question('Enter device id: ', (id) => {
            rl.question('Enter owner: ', (owner) => {
                rl.question('Enter device type: ', (deviceType) => {
                    rl.question('Enter public key: ', (publicKey) => {
                        rl.question('Enter certificate: ', (certificate) => {
                            rl.question('Enter JWT token: ', (token) => {
                                createDevice(id, owner, deviceType, publicKey, certificate, token);
                                rl.close();
                            });
                        });
                    });
                });
            });
        });
    } else if (command === 'get') {
        rl.question('Enter device id: ', (id) => {
            rl.question('Enter JWT token: ', (token) => {
                getDevice(id, token);
                rl.close();
            });
        });
    } else {
        console.log('Invalid command');
        rl.close();
    }
});
```

Figure 12: CLI setup

7 JWT Token Generation



JWT Token
JWT Expired time in seconds
Refresh token Expired time in seconds

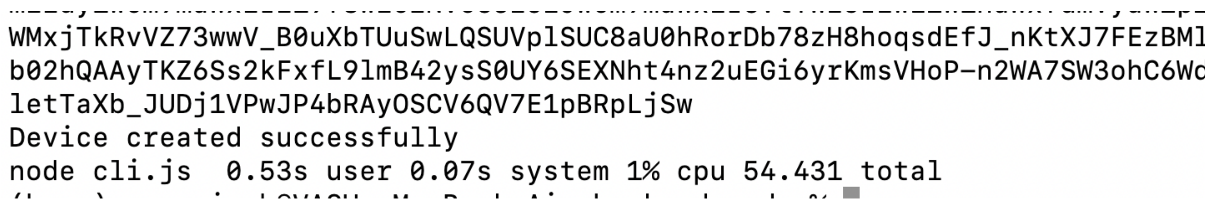
```
(base) vasusingh@VASUs-MacBook-Air project % curl --data "client_id=Blockchain-client" --data "client_secret=r6JIRNinrbKD607GRvEJSb084w2KZTIt" --data "usernam
e=vasusingh" --data "password=Wasu@9807" --data "grant_type=password" "http://localhost:8080/realms/IoTrealm/protocol/openid-connect/token"
{"access_token":"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJqVE1JaXR6OXk2WVZNeWRHZTJCSmFyNThkNl91TExpU0FzOFRTZXNfYlFBIn0.eyJleHAiOjE3MjMxNDQ3MzUsImlhd
CI6MTcyMzE0NDQzNSwianRpIjoiYTRhMTg3YTEtMzc3NC00MTg5LWEyY2YtZDgzMTBjNjUzNDlkIiwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdDo4MDgwL3JlYWxtcy9Jb1RyZWFsbSIsImF1ZCI6ImFjY291bnQ
iLCJzdWIiOiJkZDYzMTJmOS05N2Q3LTRlMTktOGIxMS1iMGMzYzljMzA2ZDQiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJCbG9ja2NoYWluLWNsaWVudCIsInNpZCI6ImZmNjBlZDAzLWQ5NDctNDNjMy04NDdmL
WI2Y2Q2NjA2OWIyZCIsImFjciI6IjEiLCJhbGxvd2VkLW9yaWdpbnMiOlsiaHR0cDovL2xvY2FsaG9zdDo4MDgwLyJdLCJyZWFsbV9hY2Nlc3MiOnsicm9sZXMiOlsiZGVmYXVsdC1yb2xlcy1pb3RyZWFsbSI
sIm9mZmxpbmVfYWNjZXNzIiwidW1hX2F1dGhvcml6YXRpb24iXX0sInJlc291cmNlX2FjY2VzcyI6eyJCbG9ja2NoYWluLWNsaWVudCI6eyJyb2xlcyI6WyJ1bWFfcHJvdGVjdGlvbiJdfSwiYWNjb3VudCI6e
yJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsIm1hbmFnZS1hY2NvdW50LWxpbmtzIiwidmlldy1wcm9maWxlIl19fSwic2NvcGUiOiJwcm9maWxlIGVtYWlsIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsInByZWZ
lcnJlZF91c2VybmFtZSI6InZhc3VzaW5naCJ9.oMaeMd-vUtgv4x_uh1jjeNK4LiLN-SiR27D84bnVnbI8oz3fhK50IbJDacZ_lIDOr9MhlH2OXjvWlViwj9_b68kVNVgqErI5n_z7kW6HbLcjHut3A7VAatRJ
c0AIJhYe_JzGnGp-L4sGWVWEnANdcpyGM8EqRwIfXmLb5PNpWRVLxfQE_6FRQRw_HIFTl4uIdrxvOZAKmH55nkeG8imk-x0Uyb9RRqZH3_ZORBxSidX9SA86rxjytxO-_AnpH9fKS0NsJjXs7TUlqa9D2HR3oI
Uh1pPDTsPMELDsEhB6xN2Uf9AF0TK0Ef0HRbMjgbkiwPSlyuaRxHCi5ss2oMACzg","expires_in":300,"refresh_expires_in":1800,"refresh_token":"eyJhbGciOiJIUzUxMiIsInR5cCIgOiAi
SldUIiwia2lkIiA6ICI5ZjU2YWQ3Mi05OWY2LTRjNzgtYjcxZC01ZjdlYzBhNmE3YzgifQ.eyJleHAiOjE3MjMxNDYyMzUsImlhdCI6MTcyMzE0NDQzNSwianRpIjoiZmM0MDdiOTItMDQxMi00MzkyLTgzMzg
tZTI3MmRiMTg4OGUwIiwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdDo4MDgwL3JlYWxtcy9Jb1RyZWFsbSIsImF1ZCI6Imh0dHA6Ly9sb2NhbGhvc3Q6ODA4MC9yZWFsbXMvSW9UcmVhbG0iLCJzdWIiOiJkZDYzM
TJmOS05N2Q3LTRlMTktOGIxMS1iMGMzYzljMzA2ZDQiLCJ0eXAiOiJSZWZyZXNoIiwiYXpwIjoiQmxvY2tjaGFpbi1jbGllbnQiLCJzaWQiOiJmZjYwZWQwMy1kOTQ3LTQzYzMtODQ3Zi1iNmNkNjYwNjliMmQ
iLCJzY29wZSI6ImJhc2ljIHByb2ZpbGUgd2ViLW9yaWdpbnMgYWNyIHJvbGVzIGVtYWlsIn0.n-BEoPEZKamkUPa2SHcTuJMBeQpIfnJFo02y-1LQ_inhaoJWS35voM4Rk9RI-pKOwZCaMK1iEvncLbAG0kU8j
Q","token_type":"Bearer","not-before-policy":0,"session_state":"ff60ed03-d947-43c3-847f-b6cd66069b2d","scope":"profile email"}%
```

Figure 13: JWT Token generation

8 Evaluation

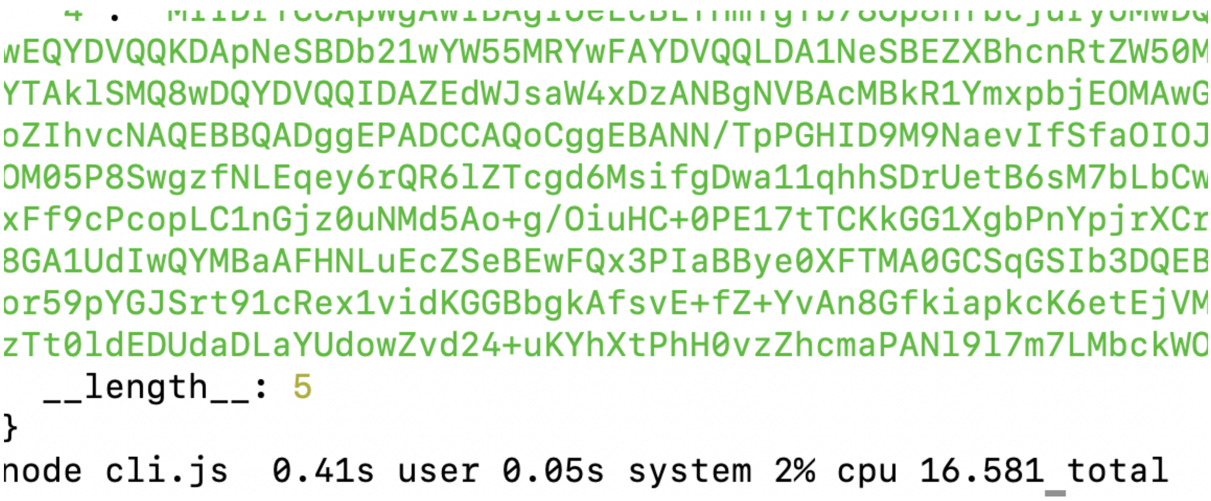
8.1 Latency

Measured the latency using the **time** command while running your CLI.js scripts.



```
WMxjTkRvVZ73wwV_B0uXbTUuSwLQSUVplSUC8aU0hRorDb78zH8hoqsdEfJ_nKtXJ7FEzBMl
b02hQAAyTKZ6Ss2kFxfL9lmB42ysS0UY6SEXNht4nz2uEGi6yrKmsVHoP-n2WA7SW3ohC6Wc
letTaXb_JUDj1VPwJP4bRAyOSCV6QV7E1pBRpLjSw
Device created successfully
node cli.js  0.53s user 0.07s system 1% cpu 54.431 total
```

Figure 14: Create device with time output



```
wEQYDVQQKDApNeSBDb21wYW55MRYwFAYDVQQLDA1NeSBEZXBhcnRtZW50M
YTAklSMQ8wDQYDVQQIDAZEdWJsaW4xDzANBgNVBAcMBkR1YmxpbjEOMAwG
oZIhvcNAQEBBQADggEPADCCAQoCggEBANN/TpPGHID9M9NaevIfSfaOIOJ
OM05P8SwgzfNLEqey6rQR6lZTcgd6MsifgDwa11qhhSDrUetB6sM7bLbCw
xFf9cPcopLC1nGjz0uNMd5Ao+g/OiuHC+0PE17tTCKkGG1XgbPnYpjrXCr
8GA1UdIwQYMBaAFHNLuEcZSeBEwFQx3PIaBBye0XFTMA0GCSqGSIb3DQEB
or59pYGJSrt91cRex1vidKGGBbgkAfsvE+fZ+YvAn8GfkiapkcK6etEjVM
zTt0ldEDUdaDLaYUdowZvd24+uKYhXtPhH0vzZhcmaPANl9l7m7LMbckWO
  __length__: 5
}
node cli.js  0.41s user 0.05s system 2% cpu 16.581 total
```

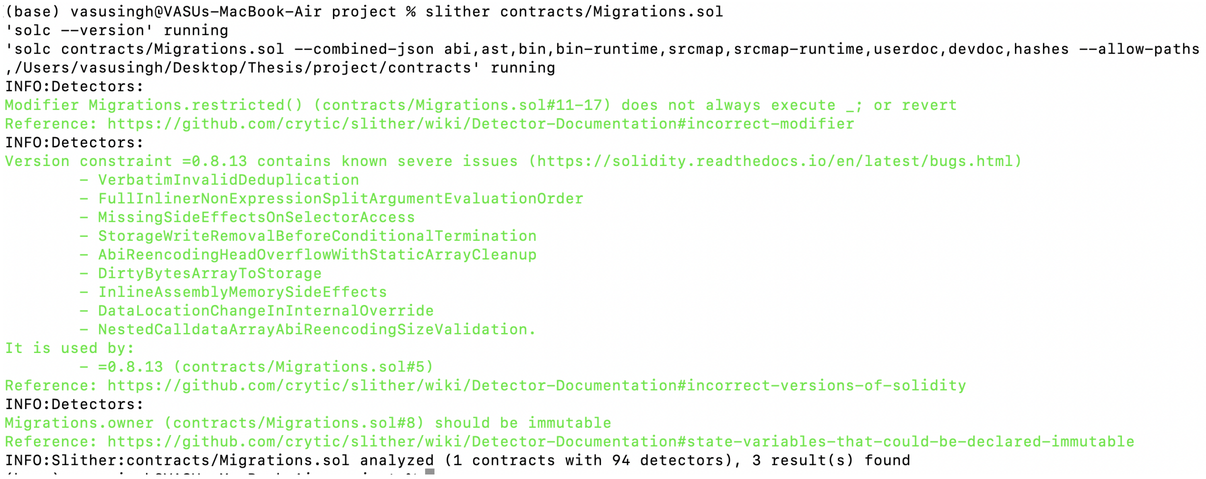
Figure 15: Get device with time output

8.2 Slither Security Audit

Installed slither using below command-

pip install slither-analyzer[8]

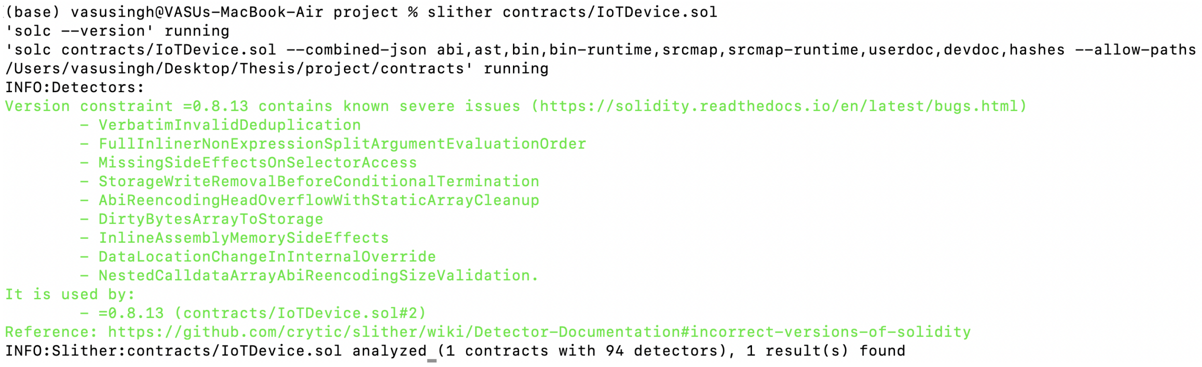
Tested both solidity scripts for the audited using Slither.



```
(base) vasusingh@VASUs-MacBook-Air project % slither contracts/Migrations.sol
'solc --version' running
'solc contracts/Migrations.sol --combined-json abi,ast,bin,bin-runtime,srcmap,srcmap-runtime,userdoc,devdoc,hashes --allow-paths
,/Users/vasusingh/Desktop/Thesis/project/contracts' running
INFO:Detectors:
Modifier Migrations.restricted() (contracts/Migrations.sol#11-17) does not always execute _; or revert
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier
INFO:Detectors:
Version constraint =0.8.13 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - VerbatimInvalidDeduplication
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess
        - StorageWriteRemovalBeforeConditionalTermination
        - AbiReencodingHeadOverflowWithStaticArrayCleanup
        - DirtyBytesArrayToStorage
        - InlineAssemblyMemorySideEffects
        - DataLocationChangeInInternalOverride
        - NestedCalldataArrayAbiReencodingSizeValidation.
It is used by:
        - =0.8.13 (contracts/Migrations.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Migrations.owner (contracts/Migrations.sol#8) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:contracts/Migrations.sol analyzed (1 contracts with 94 detectors), 3 result(s) found
```

Figure 16: Slither output for Migration.sol

[8] https://pypi.org/project/slither-analyzer/0.8.3/

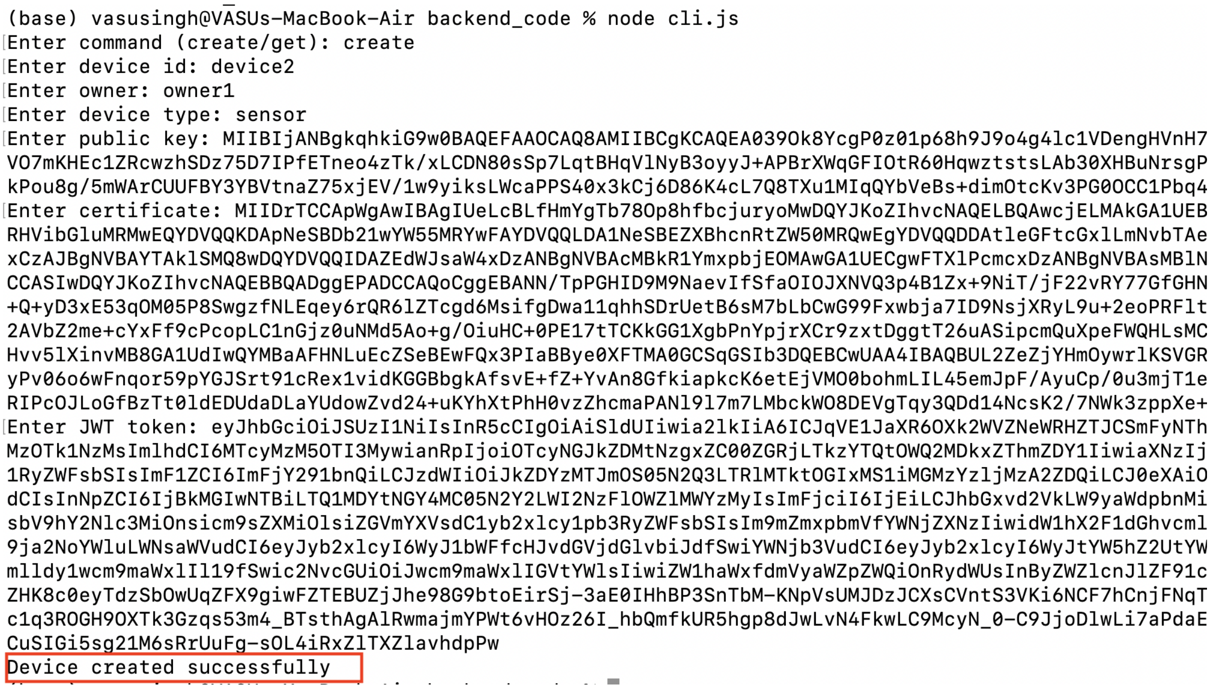


```
(base) vasusingh@VASUs-MacBook-Air project % slither contracts/IoTDevice.sol
'solc --version' running
'solc contracts/IoTDevice.sol --combined-json abi,ast,bin,bin-runtime,srcmap,srcmap-runtime,userdoc,devdoc,hashes --allow-paths
/Users/vasusingh/Desktop/Thesis/project/contracts' running
INFO:Detectors:
Version constraint =0.8.13 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - VerbatimInvalidDeduplication
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess
        - StorageWriteRemovalBeforeConditionalTermination
        - AbiReencodingHeadOverflowWithStaticArrayCleanup
        - DirtyBytesArrayToStorage
        - InlineAssemblyMemorySideEffects
        - DataLocationChangeInInternalOverride
        - NestedCalldataArrayAbiReencodingSizeValidation.
It is used by:
        - =0.8.13 (contracts/IoTDevice.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Slither:contracts/IoTDevice.sol analyzed (1 contracts with 94 detectors), 1 result(s) found
```

Figure 17: Slither output for IoTDevice.sol

8.3 JWT Token Validation

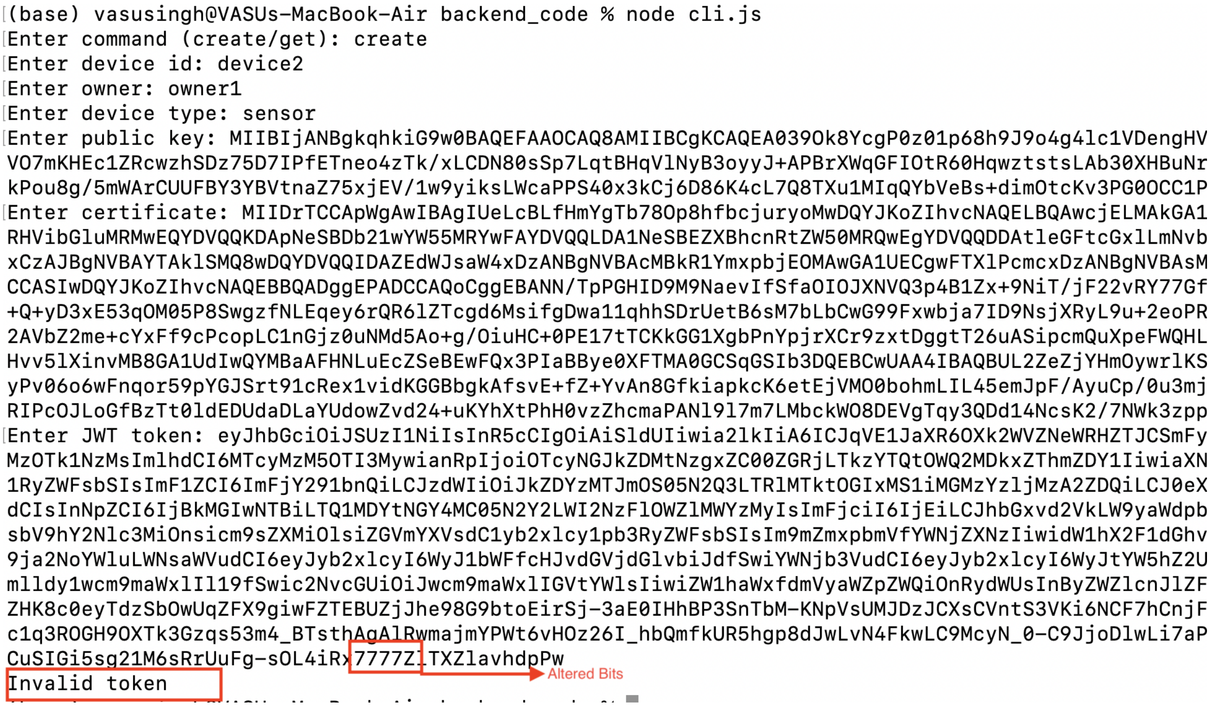
Valid Token



```
(base) vasusingh@VASUs-MacBook-Air backend_code % node cli.js
Enter command (create/get): create
Enter device id: device2
Enter owner: owner1
Enter device type: sensor
Enter public key: MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA039Ok8YcgP0z01p68h9J9o4g4lc1VDengHVnH7
VO7mKHEc1ZRcwzhSDz75D7IPfETneo4zTk/xLCDN80sSp7LqtBHqVlNyB3oyyJ+APBrXWqGFIOtR60HqwztstsLAb30XHBuNrsgP
kPou8g/5mWArCUUFBY3YBVtnaZ75xjEV/1w9yiksLWcaPPS40x3kCj6D86K4cL7Q8TXu1MIqQYbVeBs+dimOtcKv3PG0OCC1Pbq4
Enter certificate: MIIDrTCCApWgAwIBAgIUeLcBLfHmYgTb78Op8hfbcjuryoMwDQYJKoZIhvcNAQELBQAwcjELMAkGA1UEB
RHVibGluMRMwEQYDVQQKDApNeSBDb21wYW55MRYwFAYDVQQLDA1NeSBEZXBhcnRtZW50MRQwEgYDVQQDDAtleGFtcGxlLmNvbTAe
xCzAJBgNVBAYTAklSMQ8wDQYDVQQIDAZEdWJsaW4xDzANBgNVBAcMBkR1YmxpbjEOMAwGA1UECgwFTXlPcmcxDzANBgNVBAsMBlN
CCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANN/TpPGHID9M9NaevIfSfaOIOJXNVQ3p4B1Zx+9NiT/jF22vRY77GfGHN
+Q+yD3xE53qOM05P8SwgzfNLEqey6rQR6lZTcgd6MsifgDwa11qhhSDrUetB6sM7bLbCwG99Fxwbja7ID9NsjXRyL9u+2eoPRFlt
2AVbZ2me+cYxFf9cPcopLC1nGjz0uNMd5Ao+g/OiuHC+0PE17tTCKkGG1XgbPnYpjrXCr9zxtDggtT26uASipcmQuXpeFWQHLsMC
Hvv5lXinvMB8GA1UdIwQYMBaAFHNLuEcZSeBEwFQx3PIaBBye0XFTMA0GCSqGSIb3DQEBCwUAA4IBAQBUL2ZeZjYHmOywrlKSVGR
yPv06o6wFnqor59pYGJSrt91cRex1vidKGGBbgkAfsvE+fZ+YvAn8GfkiapkcK6etEjVMO0bohmLIL45emJpF/AyuCp/0u3mjT1e
RIPcOJLoGfBzTt0ldEDUdaDLaYUdowZvd24+uKYhXtPhH0vzZhcmaPANl9l7m7LMbckWO8DEVgTqy3QDd14NcsK2/7NWk3zppXe+
Enter JWT token: eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJqVE1JaXR6OXk2WVZNeWRHZTJCSmFyNTh
MzOTk1NzMsImlhdCI6MTcyMzM5OTI3MywianRpIjoiOTcyNGJkZDMtNzgxZC00ZGRjLTkzYTQtOWQ2MDkxZThmZDY1IiwiaXNzIj
1RyZWFsbSIsImF1ZCI6ImFjY291bnQiLCJzdWIiOiJkZDYzMTJmOS05N2Q3LTRlMTktOGIxMS1iMGMzYzljMzA2ZDQiLCJ0eXAiO
dCIsInNpZCI6IjBkMGIwNTBiLTQ1MDYtNGY4MC05N2Y2LWI2NzFlOWZlMWYzMyIsImFjciI6IjEiLCJhbGxvd2VkLW9yaWdpbnMi
sbV9hY2Nlc3MiOnsicm9sZXMiOlsiZGVmYXVsdC1yb2xlcy1pb3RyZWFsbSIsIm9mZmxpbmVfYWNjZXNzIiwidW1hX2F1dGhvcml
9ja2NoYWluLWNsaWVudCI6eyJyb2xlcyI6WyJ1bWFfcHJvdGVjdGlvbiJdfSwiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYW
mlldy1wcm9maWxlIl19fSwic2NvcGUiOiJwcm9maWxlIGVtYWlsIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsInByZWZlcnJlZF91c
ZHK8c0eyTdzSbOwUqZFX9giwFZTEBUZjJhe98G9btoEirSj-3aE0IHhBP3SnTbM-KNpVsUMJDzJCXsCVntS3VKi6NCF7hCnjFNqT
c1q3ROGH9OXTk3Gzqs53m4_BTsthAgAlRwmajmYPWt6vHOz26I_hbQmfkUR5hgp8dJwLvN4FkwLC9McyN_0-C9JjoDlwLi7aPdaE
CuSIGi5sg21M6sRrUuFg-sOL4iRxZlTXZlavhdpPw
Device created successfully
```

Figure 18: Creating device with valid token

Tampered Token

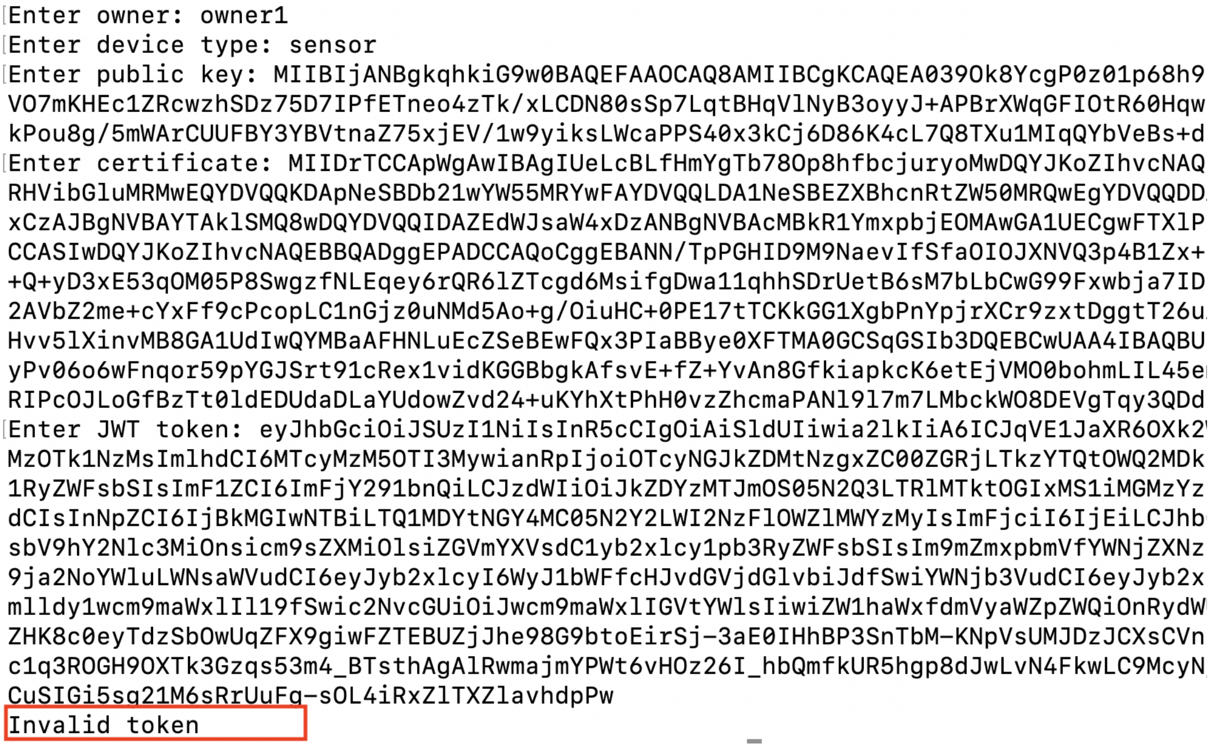


```
(base) vasusingh@VASUs-MacBook-Air backend_code % node cli.js
Enter command (create/get): create
Enter device id: device2
Enter owner: owner1
Enter device type: sensor
Enter public key: MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA039Ok8YcgP0z01p68h9J9o4g4lc1VDengHV
VO7mKHEc1ZRcwzhSDz75D7IPfETneo4zTk/xLCDN80sSp7LqtBHqVlNyB3oyyJ+APBrXWqGFIOtR60HqwztstsLAb30XHBuNr
kPou8g/5mWArCUUFBY3YBVtnaZ75xjEV/1w9yiksLWcaPPS40x3kCj6D86K4cL7Q8TXu1MIqQYbVeBs+dimOtcKv3PG0OCC1P
Enter certificate: MIIDrTCCApWgAwIBAgIUeLcBLfHmYgTb78Op8hfbcjuryoMwDQYJKoZIhvcNAQELBQAwcjELMAkGA1
RHVibGluMRMwEQYDVQQKDApNeSBDb21wYW55MRYwFAYDVQQLDA1NeSBEZXBhcnRtZW50MRQwEgYDVQQDDAtleGFtcGxlLmNvb
xCzAJBgNVBAYTAklSMQ8wDQYDVQQIDAZEdWJsaW4xDzANBgNVBAcMBkR1YmxpbjEOMAwGA1UECgwFTXlPcmcxDzANBgNVBAsM
CCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANN/TpPGHID9M9NaevIfSfaOIOJXNVQ3p4B1Zx+9NiT/jF22vRY77Gf
+Q+yD3xE53qOM05P8SwgzfNLEqey6rQR6lZTcgd6MsifgDwa11qhhSDrUetB6sM7bLbCwG99Fxwbja7ID9NsjXRyL9u+2eoPR
2AVbZ2me+cYxFf9cPcopLC1nGjz0uNMd5Ao+g/OiuHC+0PE17tTCKkGG1XgbPnYpjrXCr9zxtDggtT26uASipcmQuXpeFWQHL
Hvv5lXinvMB8GA1UdIwQYMBaAFHNLuEcZSeBEwFQx3PIaBBye0XFTMA0GCSqGSIb3DQEBCwUAA4IBAQBUL2ZeZjYHmOywrlKS
yPv06o6wFnqor59pYGJSrt91cRex1vidKGGBbgkAfsvE+fZ+YvAn8GfkiapkcK6etEjVMO0bohmLIL45emJpF/AyuCp/0u3mj
RIPcOJLoGfBzTt0ldEDUdaDLaYUdowZvd24+uKYhXtPhH0vzZhcmaPANl9l7m7LMbckWO8DEVgTqy3QDd14NcsK2/7NWk3zpp
Enter JWT token: eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJqVE1JaXR6OXk2WVZNeWRHZTJCSmFy
MzOTk1NzMsImlhdCI6MTcyMzM5OTI3MywianRpIjoiOTcyNGJkZDMtNzgxZC00ZGRjLTkzYTQtOWQ2MDkxZThmZDY1IiwiaXN
1RyZWFsbSIsImF1ZCI6ImFjY291bnQiLCJzdWIiOiJkZDYzMTJmOS05N2Q3LTRlMTktOGIxMS1iMGMzYzljMzA2ZDQiLCJ0eX
dCIsInNpZCI6IjBkMGIwNTBiLTQ1MDYtNGY4MC05N2Y2LWI2NzFlOWZlMWYzMyIsImFjciI6IjEiLCJhbGxvd2VkLW9yaWdpb
sbV9hY2Nlc3MiOnsicm9sZXMiOlsiZGVmYXVsdC1yb2xlcy1pb3RyZWFsbSIsIm9mZmxpbmVfYWNjZXNzIiwidW1hX2F1dGhv
9ja2NoYWluLWNsaWVudCI6eyJyb2xlcyI6WyJ1bWFfcHJvdGVjdGlvbiJdfSwiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2U
mlldy1wcm9maWxlIl19fSwic2NvcGUiOiJwcm9maWxlIGVtYWlsIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsInByZWZlcnJlZF
ZHK8c0eyTdzSbOwUqZFX9giwFZTEBUZjJhe98G9btoEirSj-3aE0IHhBP3SnTbM-KNpVsUMJDzJCXsCVntS3VKi6NCF7hCnjF
c1q3ROGH9OXTk3Gzqs53m4_BTsthAgAlRwmajmYPWt6vHOz26I_hbQmfkUR5hgp8dJwLvN4FkwLC9McyN_0-C9JjoDlwLi7aP
CuSIGi5sg21M6sRrUuFg-sOL4iRx7777ZlTXZlavhdpPw
Invalid token
```

Altered Bits

Figure 19: Creating device with tampered token

Expired Token

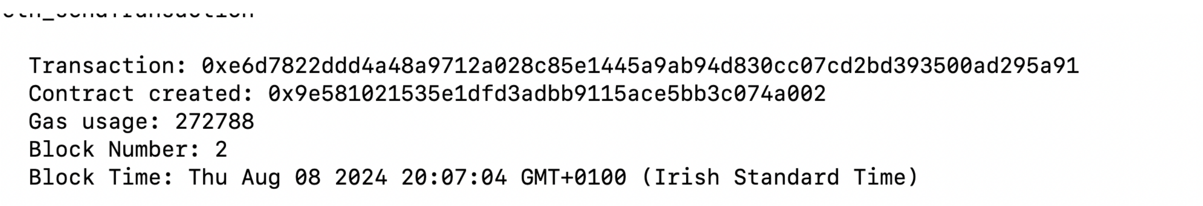


```
Enter owner: owner1
Enter device type: sensor
Enter public key: MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA039Ok8YcgP0z01p68h9
VO7mKHEc1ZRcwzhSDz75D7IPfETneo4zTk/xLCDN80sSp7LqtBHqVlNyB3oyyJ+APBrXWqGFIOtR60Hqw
kPou8g/5mWArCUUFBY3YBVtnaZ75xjEV/1w9yiksLWcaPPS40x3kCj6D86K4cL7Q8TXu1MIqQYbVeBs+d
Enter certificate: MIIDrTCCApWgAwIBAgIUeLcBLfHmYgTb78Op8hfbcjuryoMwDQYJKoZIhvcNAQ
RHVibGluMRMwEQYDVQQKDApNeSBDb21wYW55MRYwFAYDVQQLDA1NeSBEZXBhcnRtZW50MRQwEgYDVQQDD
xCzAJBgNVBAYTAklSMQ8wDQYDVQQIDAZEdWJsaW4xDzANBgNVBAcMBkR1YmxpbjEOMAwGA1UECgwFTXlP
CCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANN/TpPGHID9M9NaevIfSfaOIOJXNVQ3p4B1Zx+
+Q+yD3xE53qOM05P8SwgzfNLEqey6rQR6lZTcgd6MsifgDwa11qhhSDrUetB6sM7bLbCwG99Fxwbja7ID
2AVbZ2me+cYxFf9cPcopLC1nGjz0uNMd5Ao+g/OiuHC+0PE17tTCKkGG1XgbPnYpjrXCr9zxtDggtT26u
Hvv5lXinvMB8GA1UdIwQYMBaAFHNLuEcZSeBEwFQx3PIaBBye0XFTMA0GCSqGSIb3DQEBCwUAA4IBAQBU
yPv06o6wFnqor59pYGJSrt91cRex1vidKGGBbgkAfsvE+fZ+YvAn8GfkiapkcK6etEjVMO0bohmLIL45e
RIPcOJLoGfBzTt0ldEDUdaDLaYUdowZvd24+uKYhXtPhH0vzZhcmaPANl9l7m7LMbckWO8DEVgTqy3QDd
Enter JWT token: eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJqVE1JaXR6OXk2
MzOTk1NzMsImlhdCI6MTcyMzM5OTI3MywianRpIjoiOTcyNGJkZDMtNzgxZC00ZGRjLTkzYTQtOWQ2MDk
1RyZWFsbSIsImF1ZCI6ImFjY291bnQiLCJzdWIiOiJkZDYzMTJmOS05N2Q3LTRlMTktOGIxMS1iMGMzYz
dCIsInNpZCI6IjBkMGIwNTBiLTQ1MDYtNGY4MC05N2Y2LWI2NzFlOWZlMWYzMyIsImFjciI6IjEiLCJhb
sbV9hY2Nlc3MiOnsicm9sZXMiOlsiZGVmYXVsdC1yb2xlcy1pb3RyZWFsbSIsIm9mZmxpbmVfYWNjZXNz
9ja2NoYWluLWNsaWVudCI6eyJyb2xlcyI6WyJ1bWFfcHJvdGVjdGlvbiJdfSwiYWNjb3VudCI6eyJyb2x
mlldy1wcm9maWxlIl19fSwic2NvcGUiOiJwcm9maWxlIGVtYWlsIiwiZW1haWxfdmVyaWZpZWQiOnRydW
ZHK8c0eyTdzSbOwUqZFX9giwFZTEBUZjJhe98G9btoEirSj-3aE0IHhBP3SnTbM-KNpVsUMJDzJCXsCVn
c1q3ROGH9OXTk3Gzqs53m4_BTsthAgAlRwmajmYPWt6vHOz26I_hbQmfkUR5hgp8dJwLvN4FkwLC9McyN
CuSIGi5sg21M6sRrUuFg-sOL4iRxZlTXZlavhdpPw
Invalid token
```

Figure 20: Creating device with expired token

8.4 Privacy Testing

Only metadata such as transaction hash, gas usage is stored on the blockchain after device registration.



```
eth_sendTransaction

  Transaction: 0xe6d7822ddd4a48a9712a028c85e1445a9ab94d830cc07cd2bd393500ad295a91
  Contract created: 0x9e581021535e1dfd3adbb9115ace5bb3c074a002
  Gas usage: 272788
  Block Number: 2
  Block Time: Thu Aug 08 2024 20:07:04 GMT+0100 (Irish Standard Time)
```

Figure 21: Blockchain transaction