**Blockchain Workshop**

Lab Session 4

Sergi Pla

Jordi Otal

Arnau Garcia

Enric Pedrico

## 1. Local environment setup

## 2. Write and compile the Smart Contract

### • P1- Try to understand what the contract does. Can you explain it?

The Box contract is a small program for the Ethereum blockchain. Its job is to store and retrieve a number. Here’s how it works:

* It has a private variable called \_value to keep the number safe.
* There’s a function called store to save a number. Before saving, it checks that the number is greater than 0 (to avoid saving something invalid like 0 or a negative number).
* Another function, retrieve, lets you see the stored number.
* When you save a number, it sends an event called ValueChanged to tell others that the number has changed.

### • P2- Why do you think is important to validate inputs (e.g value > 0)

Validation is needed to check that the input makes sense before saving or using it. In this case, we check value > 0 in order to:

* **Avoid mistakes**: It stops invalid data from being saved, like 0 or negative numbers, which might break the program logic.
* **Save money**: On Ethereum, every action costs money (gas). If you don’t check inputs, someone could waste gas for something useless.
* **Make it secure**: Without validation, someone could try to break the system by sending invalid data, creating unexpected problems.

## 3. Deploy locally

## 4. Interact with the Smart Contract

### • P3- Write down the block parameters you obtained.

ContractTransactionResponse {

provider: HardhatEthersProvider {

\_hardhatProvider: LazyInitializationProviderAdapter {

\_providerFactory: [AsyncFunction (anonymous)],

\_emitter: [EventEmitter],

\_initializingPromise: [Promise],

provider: [BackwardsCompatibilityProviderAdapter]

},

\_networkName: 'localhost',

\_blockListeners: [],

\_transactionHashListeners: Map(0) {},

\_eventListeners: []

},

**blockNumber: 4,**

**blockHash: '0x7b05a0d7737d57282b773883039c5d016fa619098689d0cf38f1a289971357a1',**

index: undefined,

hash: '0xe92eca435ba34cf32f41610807b08de15d6fe0058e5230406d8386d9447a2791',

type: 2,

to: '0x5FbDB2315678afecb367f032d93F642f64180aa3',

from: '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266',

nonce: 3,

gasLimit: 30000000n,

gasPrice: 743644643n,

maxPriorityFeePerGas: 178327267n,

maxFeePerGas: 743644643n,

maxFeePerBlobGas: null,

data: '0x6057361d0000000000000000000000000000000000000000000000000000000000000021',

value: 0n,

chainId: 31337n,

signature: Signature { r: "0x8dce1cc24b74c92ddb4e09c7adeb583c42f7b31caecc1594fbc71dff5b08ea6e", s: "0x1e9bb528b74e3e0ec78ba9f66caeb1d3ed8a4567198636e783e0b8cc557bfec4", yParity: 1, networkV: null },

accessList: [],

blobVersionedHashes: null

}



### • P4- Explain what Gas is and why transactions in Ethereum consume Gas. Which type of attack does it help to mitigate?

Gas is a fee you pay for running a program or saving data on Ethereum. It would be like paying for computer power. The more complex your program, the more gas you pay.

Gas makes sure people don’t spam the blockchain with too many transactions or programs that run forever. It also pays the validators (the people running the network) for their work.

It prevents Denial of Service (DoS) attacks, where someone might try to overload the network with useless or infinite tasks. Gas puts a limit on how much computing power they can use.

## 5. Deploy to a public Testnet

## 6. Etherscan

### • P5- Include the contract address in the Lab report

Sepolia contract address: **0x1801c1A4c7f7Bd59CD4CADB514f0455801b02e80**

## 7. Interact with the Smart Contract with Etherscan and MetaMask

### • P6- How much SepoliaEth will the transaction cost? Explain how transaction cost is computed.

The transaction costs 0,00224125 SepoliaEth. The transaction is computed as the product of the gas used and the gas price, Transaction Cost = Gas Used \* Gas Price.

### • P7- Is the transaction persisted immediately? Why? Explain the consensus protocol currently used in Ethereum networks.

The transaction is not persisted immediately because it needs to be verified / validated.

Ethereum currently uses the Proof of Stake consensus protocol which makes sure that a transaction is validated and added to the blockchain only after it has been confirmed by the validators.

These validators are responsible for checking and approving transactions. They have to deposit some ETH as a commitment to act honestly in the network.

When a transaction is sent, it is first added to a block proposed by one of the validators. Other validators then check the block and confirm that everything is correct. Only after enough validators agree, the block is added to the blockchain and the transaction is saved.

The prioritization of the transactions depend on the gas price offered. The ones with higher fees are processed faster and the ones with lower fees will take more time to be included.

7.2 Does it return the same value you wrote before?

Yes, it does return the value 33.

### • P8- Retrieve the transaction information associated to the Store function call. Also, write down the information of the block where the transaction is registered. Add some screenshots to the report.

| **Contract details:** |
| --- |
|  |
| **Transaction details:** |
|  |
|  |
| **Block details:** |
|  |

### • P9- Do you think 51% attacks are possible in Ethereum mainnet today? Justify your answer.

No, a 51% attack is almost impossible in Ethereum today because of the properties of the blockchain:

* **Ethereum uses Proof of Stake (PoS)**: In PoS, you need to own 51% of all the staked Ether (ETH) to control the network. This would cost billions of euros, making it nearly impossible.
* **Economic logic**: If someone tried to attack, they would lose a lot of money because attacking destroys the value of ETH.
* **Punishment system**: If a validator tries to cheat, they lose part of their staked Ether (this is called “slashing”).
* **Decentralized network**: Ethereum has many validators around the world, so no single group can easily take control.