Learning module 5: Smart Contracts

Description

The term “Smart Contracts” is used with two separate interpretations:

- “A Smart Contract is a collection of code (its functions) and data (its state) that resides at a specific address on the blockchain.”
- “A Smart Contract (as a Contract) is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract.”

Learning Module 5 investigates design issues of blockchain-based applications (Smart Contracts) that can automatically execute the terms of a contract, focusing on both technological and business aspects of the subject.

The module begins with an introduction to the concept of Smart Contract and a description of its historical development. An overview of programming languages used for development of Smart Contracts for blockchain systems and execution environments is presented, followed by a detailed description of the Ethereum Virtual Machine and a thematic lab on development of Smart Contract for the Ethereum platform using the Solidity programming language. Oracles, a special category of Smart Contracts that communicate with trusted entities, are described in detail. Computational costs and security issues of Smart Contract programming are discussed, and best practices adopted to mitigate problems and risks presented. Finally, the Smart Contracts could be used as a contract, the main regulatory and legal issues about the Smart Contracts should are identified and described.

Dependencies

This learning module has the following prerequisites:

- Consensus mechanisms (LM3)
- Digital Signatures (LM4)

Learning Objectives

- To define and explain basic properties of a Smart Contract.
- To present different programming languages used to develop Smart Contracts and their execution environments.
- To describe the Ethereum blockchain and the Ethereum Virtual Machine.
- To explain how to use the Solidity language to develop a Smart Contract for the Ethereum Blockchain.
- To present the concept of Oracles their basic functioning principles.
- To discuss the computational costs of deploying and executing a Smart Contract and the role of Gas on the Ethereum Blockchain.
- To discuss security issues related to the development and use of Smart Contracts.
- To present regulation frameworks affecting the use of Smart Contracts.
- To present legal issues related to the use of Smart Contracts in a blockchain-based system.

Learning Outcomes

- **Describe** the basic concepts of Smart Contracts.
- **Recognize** different Smart Contracts' programming languages and their execution environments.
- **Identify** the key features of different Smart Contracts' programming languages.
- **Describe** the Ethereum Blockchain.
- **Explain** how the Ethereum Virtual Machine works.
- **Implement** Smart Contracts in Ethereum using Solidity.
- **Examine** the computational cost of deploying and using a Smart Contract in Ethereum.
- **Demonstrate** Smart Contracts' use cases in multiple domains.
- **Understand** current technology's restrictions and limitations.
- **Identify** issues related to the development of Smart Contracts.
- **Examine** advantages and disadvantages of using a Smart Contract.
- **Identify** concerns about security, stability and cost of a Smart Contract.
- **Assess** whether a Smart Contract solution is suitable for the problem under study.
- **Assess** how and when to apply Smart Contracts in real-life applications.
- **Implement** Smart Contracts in real-life applications.

Syllabus

1. Introduction to Smart Contracts
   1.1. Basic definitions of Smart Contract
   1.2. Where is a Smart Contract executed?
   1.3. Understanding the Virtual Machine of a Blockchain
   1.4. Viewing the source code of a Smart Contract
   1.5. Useful Terminology & Concepts n Smart Contracts
   1.6. Good practices in the usage Smart Contracts
2. Introduction to Ethereum Smart Contracts and the Ethereum Virtual Machine (EVM)
   2.1. Prevalence of the Ethereum blockchain in Smart Contract development
   2.2. Ether and Gas: The cost of running a Smart Contract
   2.3. Getting a taste: Working examples of Ethereum Smart Contract
   2.4. Multiple Smart Contracts working together
3. Trust, Security & Efficiency of Smart Contracts "in the field"
   3.1. Market impact & scientific innovation of Smart Contract
   3.2. Understanding Trust
   3.3. Constructing Future-resistant systems using Smart Contracts
   3.4. The double importance of Security in Smart Contracts
   3.5. Novelty of Smart Contracts
   3.6. Famous smart-contract-related hacks and scandals
4. Smart Contract programming languages and execution environments
   4.1. Writing Smart Contracts
   4.2. The workflow of developing a Smart Contract
   4.3. The programmer's responsibility when writing Smart Contract
   4.4. Cost of running smart Contracts
   4.5. Presentation of different Smart Contract development platforms and their differences
   4.6. Picking the right blockchain for you: Where to develop your Smart Contract

5. Developing Smart Contracts on the Ethereum blockchain
   - 5.1. The Solidity language: An object-oriented, high-level language for implementing smart contract
   - 5.2. Designing a Solidity Smart Contract
   - 5.3. Working examples of Solidity Smart Contracts
6. Oracle services
   - 6.1. Definition of a blockchain Oracle: What they are, what they do
   - 6.2. Injecting verifiable data on the blockchain
   - 6.3. Centralization chokepoints introduced by Oracle services
   - 6.4. Decentralized Oracle Service
7. Security issues in Smart Contracts & repercussions
   - 7.1. Shifting from Trust-in-People to Trust-in-Code
   - 7.2. Data permanence in Smart Contract transaction
   - 7.3. Selective-Obscurity in Smart Contract data
   - 7.4. Quantum readiness of your Smart Contract
   - 7.5. Developer Responsibility when writing a Smart Contract
   - 7.6. Smart Contract End of Life: staying agile
   - 7.7. Forks (Hard & Soft) and community security counter measures against Smart Contract exploit
8. Real world scenarios of Smart Contract use (Use-cases)
   - 8.1. Smart Contracts put to work: Real world use-cases
   - 8.2. The Ant in the Anthill paradigm: Developers looking for the best platform to write Smart Contracts on
   - 8.3. Starting opportunities: Dev contests, bounties, funding for Smart Contract projects
9. Smart Contracts in Business
   - 9.1. How Smart Contracts elevate the EU/Int'l business landscape
   - 9.2. A creative toy problem: Agricultural Supply Chain Management through Smart Contracts
   - 9.3. Funding opportunities & tenders to support the implementation of Smart Contract systems in the EU
   - 9.4. Legal Framework surrounding the use of Smart Contracts
   - 9.5. Major Hacks & Attacks: Dangers of using Open Smart Contract