



aa84-d4e1-e279-71b9

**LEVEL 4**

**BLOCKCHAIN**

**Lecturer Guide**

**Modification History**

|  |  |  |
| --- | --- | --- |
| Version | Date | Revision Description |
| V1.0 | May 2011 | For Release |
| V1.1 | November 2011 | Assessment methodology updated |
| V1.2 | September 2018 | Updated TQT |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

© NCC Education Limited, 2018 All Rights Reserved

The copyright in this document is vested in NCC Education Limited. The document must not be reproduced by any means, in whole or in part, or used for manufacturing purposes, except with the prior written permission of NCC Education Limited and then only on condition that this notice is included in any such reproduction.

#### Published by: NCC Education Limited, The Towers, Towers Business Park, Wilmslow Road, Didsbury, Manchester M20 2EZ, UK.

Tel: +44 (0) 161 438 6200 Fax: +44 (0) 161 438 6240 Email: [info@nccedu.com](mailto:info@nccedu.com) [http://www.nccedu.com](http://www.nccedu.com/)

**CONTENTS**

[1. Module Overview and Objectives 10](#_Toc171688462)

[2. Learning Outcomes and Assessment Criteria 10](#_Toc171688463)

[3. Syllabus 12](#_Toc171688464)

[4. Related National Occupational Standards 14](#_Toc171688465)

[5. Resources 14](#_Toc171688466)

[6. Pedagogic Approach 15](#_Toc171688467)

[6.1 Lectures 15](#_Toc171688468)

[6.2 Laboratory Sessions 15](#_Toc171688469)

[6.3 Private Study 15](#_Toc171688470)

[7. Assessment 15](#_Toc171688471)

[8. Further Reading List 15](#_Toc171688472)

[Topic 1: Introduction 16](#_Toc171688473)

[1.1 Learning Objectives 16](#_Toc171688474)

[1.2 Pedagogic Approach 16](#_Toc171688475)

[1.3 Timings 16](#_Toc171688476)

[1.4 Lecture Notes 17](#_Toc171688477)

[1.4.1 Guidance on the Use of the Slides 17](#_Toc171688478)

[1.4.2 Lecture 1 17](#_Toc171688479)

[Slide 4-5: 17](#_Toc171688480)

[Definition of Blockchain: Blockchain is a distributed, peer-to-peer network that allows multiple non-trusting parties to transact without a trusted intermediary, maintaining an ever-growing list of time-sequenced records. 17](#_Toc171688481)

[1.4.3 Lecture 2 17](#_Toc171688482)

[Slide 6: 17](#_Toc171688483)

[How BLOCKCHAIN technology works: Blockchain works by initiating and broadcasting transactions to a network of nodes that validate them. Verified transactions are combined into a block, which is then added to the blockchain by miners using consensus algorithms, making the transaction immutable. 17](#_Toc171688484)

[1.4.4 Lecture 3 17](#_Toc171688485)

[Slide 7-20: 17](#_Toc171688486)

[Components of BLOCKCHAIN: Blockchain components include nodes (devices connected to the network), a ledger (an immutable, distributed database), wallets (digital tools for managing cryptocurrencies), nonce (a random number for block creation), hash (a unique identifier for data), and mining (the process of validating transactions and adding them to the ledger). 17](#_Toc171688487)

[1.4.5 Lecture 4 17](#_Toc171688488)

[Slide 21-27 17](#_Toc171688489)

[Types of Blockchain: There are three types of blockchains: public (accessible to anyone), private (restricted access), and consortium (controlled by a group of organizations). 17](#_Toc171688490)

[1.4.6 Lecture 5 18](#_Toc171688491)

[Slide 28-32 18](#_Toc171688492)

[Consensus Algorithm: A consensus algorithm is a procedure used in blockchain networks to achieve agreement on the present state of the data among distributed nodes. This ensures that all participants in the network have the same data. Common consensus algorithms include Proof of Work (PoW), where miners solve complex puzzles to add a block, and Proof of Stake (PoS), where validators are chosen based on the number of coins they hold and are willing to "stake" as collateral. 18](#_Toc171688493)

[1.4.7 Lecture 5 18](#_Toc171688494)

[Slide 33-25 18](#_Toc171688495)

[1.5 Laboratory Sessions 19](#_Toc171688496)

[1.6 Private Study 20](#_Toc171688497)

[Topic 2: Cryptocurrency 22](#_Toc171688498)

[2.1 Learning Objectives 22](#_Toc171688499)

[2.2 Pedagogic Approach 22](#_Toc171688500)

[2.3 Timings 22](#_Toc171688501)

[2.4 Lecture Notes 23](#_Toc171688502)

[2.4.1 Guidance on the Use of the Slides 23](#_Toc171688503)

[2.4.2 Lecture 1 23](#_Toc171688504)

[Slide 37 23](#_Toc171688505)

[2.4.3 Lecture 2 23](#_Toc171688506)

[Slide 38-39 23](#_Toc171688507)

[2.4.4 Lecture 3 23](#_Toc171688508)

[Slide 40-41 23](#_Toc171688509)

[UTXO: UTXO refers to the amount of digital currency remaining after a cryptocurrency transaction is executed. These unspent outputs are used as inputs for future transactions, ensuring accurate and traceable accounting of funds. 23](#_Toc171688510)

[2.4.5 Lecture 4 24](#_Toc171688511)

[Slide 42-47 24](#_Toc171688512)

[Cryptocurrency Wallets: Cryptocurrency wallets are software programs that store private and public keys, allowing users to send, receive, and manage their digital assets. They come in various forms, including hot wallets (online), cold wallets (offline), hardware wallets, and paper wallets. 24](#_Toc171688513)

[2.4.6 Lecture 5 24](#_Toc171688514)

[Slide 48-51 24](#_Toc171688515)

[Altcoins: Altcoins are alternative cryptocurrencies to Bitcoin, created by forking the Bitcoin or other blockchain code. They often offer different features, improved functionality, or enhanced security compared to their predecessors. 24](#_Toc171688516)

[Tokens: Tokens are digital assets created on existing blockchains, representing either a utility (access to a service) or security (investment). They are commonly used in fundraising through Initial Coin Offerings (ICOs) and for transactions within decentralized applications. 24](#_Toc171688517)

[2.4.7 Lecture 6 24](#_Toc171688518)

[Slide 52-67 24](#_Toc171688519)

[Block Frequency: The regular interval at which new blocks are added to the blockchain, such as every ten minutes in Bitcoin . 24](#_Toc171688520)

[2.5 Laboratory Sessions 25](#_Toc171688521)

[2.6 Private Study 26](#_Toc171688522)

[Topic 3: Programming in Blockchain 29](#_Toc171688523)

[3.1 Learning Objectives 29](#_Toc171688524)

[3.2 Pedagogic Approach 29](#_Toc171688525)

[3.3 Timings 29](#_Toc171688526)

[3.4 Lecture Notes 30](#_Toc171688527)

[3.4.1 Guidance on the Use of the Slides 30](#_Toc171688528)

[3.4.2 Lecture 1 30](#_Toc171688529)

[Slide 69-75 30](#_Toc171688530)

[Smart contracts are self-executing digital contracts with the terms of the agreement directly written into lines of code. These contracts automatically enforce and execute the agreement once predefined conditions are met, ensuring trust and transparency without the need for intermediaries. 30](#_Toc171688531)

[3.4.3 Lecture 2 30](#_Toc171688532)

[Slide 76-80 30](#_Toc171688533)

[Components of Smart Contract 30](#_Toc171688534)

[The main components of a smart contract include variables (to store data), functions (to define the contract's operations), and events (to log activities). These elements work together to manage the contract's state and interactions on the blockchain. 30](#_Toc171688535)

[3.4.4 Lecture 3 30](#_Toc171688536)

[Slide 81-83 30](#_Toc171688537)

[Smart Contract Approaches 30](#_Toc171688538)

[Smart contract approaches involve different methods to create and deploy contracts, including using specific programming languages like Solidity, defining clear terms and conditions, and ensuring security measures to prevent vulnerabilities and attacks. 30](#_Toc171688539)

[3.4.5 Lecture 4 30](#_Toc171688540)

[Slide 84-88 30](#_Toc171688541)

[Data Types in Solidity 30](#_Toc171688542)

[Solidity, the primary language for Ethereum smart contracts, supports various data types including integers, booleans, strings, and arrays. These data types are used to store and manipulate information within the contract. 30](#_Toc171688543)

[3.4.6 Lecture 5 31](#_Toc171688544)

[Slide 89-91 31](#_Toc171688545)

[Functions in Solidity 31](#_Toc171688546)

[Functions in Solidity define the behavior of a smart contract. They can be public or private, allowing interaction with the contract's data. Functions can also be marked as "view" or "pure" to indicate they do not modify the contract's state. 31](#_Toc171688547)

[3.4.7 Lecture 6 31](#_Toc171688548)

[Slide 92 31](#_Toc171688549)

[Visibility Quantifiers 31](#_Toc171688550)

[Visibility quantifiers in Solidity determine who can call a function or access a variable. The main visibility types are public, private, internal, and external, each providing different levels of access control. 31](#_Toc171688551)

[3.4.8 Lecture 7 31](#_Toc171688552)

[Slide 93 31](#_Toc171688553)

[Inheritance 31](#_Toc171688554)

[Inheritance in Solidity allows one contract to inherit properties and functions from another, promoting code reusability and modularity. This enables developers to build complex systems by extending existing contracts. 31](#_Toc171688555)

[3.4.9 Lecture 8 31](#_Toc171688556)

[Slide 94 31](#_Toc171688557)

[Error Handling 31](#_Toc171688558)

[Error handling in Solidity involves using mechanisms like require(), assert(), and revert() to ensure the contract functions as intended. These tools help manage errors and validate conditions, enhancing the contract's security and reliability. 31](#_Toc171688559)

[3.5 Laboratory Sessions 32](#_Toc171688560)

[3.6 Private Study 35](#_Toc171688561)

[Topic 4: Public Blockchain 39](#_Toc171688562)

[4.1 Learning Objectives 39](#_Toc171688563)

[4.2 Pedagogic Approach 39](#_Toc171688564)

[4.3 Timings 39](#_Toc171688565)

[4.4 Lecture Notes 40](#_Toc171688566)

[4.4.1 Guidance on the Use of the Slides 40](#_Toc171688567)

[4.4.2 Lecture 1 40](#_Toc171688568)

[Slide 96-100 40](#_Toc171688569)

[Ethereum Components 40](#_Toc171688570)

[Ethereum comprises several key components, including smart contracts, decentralized applications (DApps), Ether (ETH) as its native cryptocurrency, and the Ethereum Virtual Machine (EVM) which executes smart contracts. These elements work together to create a versatile and programmable blockchain platform. 40](#_Toc171688571)

[4.4.3 Lecture 2 40](#_Toc171688572)

[Slide 101-105 40](#_Toc171688573)

[Mining in Ethereum 40](#_Toc171688574)

[Mining in Ethereum involves validating transactions and creating new blocks by solving complex cryptographic puzzles. Miners compete to solve these puzzles using a Proof of Work (PoW) consensus mechanism, which secures the network and adds verified blocks to the blockchain. 40](#_Toc171688575)

[4.4.4 Lecture 3 40](#_Toc171688576)

[Slide 106-107 40](#_Toc171688577)

[Architecture of Ethereum 40](#_Toc171688578)

[The architecture of Ethereum includes nodes, the Ethereum Virtual Machine (EVM), smart contracts, and a decentralized network. Nodes maintain the blockchain, the EVM executes code, and smart contracts automate transactions, creating a robust and programmable blockchain ecosystem. 40](#_Toc171688579)

[4.4.5 Lecture 4 40](#_Toc171688580)

[Slide 108-109 40](#_Toc171688581)

[Workflow of Ethereum 40](#_Toc171688582)

[The workflow of Ethereum involves creating and deploying smart contracts, processing transactions, and achieving consensus through mining. Transactions are initiated by users, validated by miners, and recorded on the blockchain, ensuring transparency and immutability. 40](#_Toc171688583)

[4.4.6 Lecture 5 41](#_Toc171688584)

[Slide 110-111 41](#_Toc171688585)

[Comparison between Bitcoin and Ethereum 41](#_Toc171688586)

[Bitcoin and Ethereum are both prominent public blockchains but differ in purpose and functionality. Bitcoin focuses on secure, decentralized digital currency transactions, while Ethereum provides a platform for building and executing smart contracts and decentralized applications. 41](#_Toc171688587)

[4.4.7 Lecture 6 41](#_Toc171688588)

[Slide 112-114 41](#_Toc171688589)

[Transferring Ethers Using MetaMask 41](#_Toc171688590)

[MetaMask is a browser extension that allows users to manage their Ethereum wallets and interact with the Ethereum blockchain. Users can transfer Ether by connecting MetaMask to the network, entering the recipient's address, specifying the amount, and confirming the transaction. 41](#_Toc171688591)

[4.4.8 Lecture 7 41](#_Toc171688592)

[Slide 115-116 41](#_Toc171688593)

[Ethereum Frameworks 41](#_Toc171688594)

[Ethereum frameworks such as Truffle, Hardhat, and OpenZeppelin provide tools and libraries for developing, testing, and deploying smart contracts. These frameworks simplify the development process and ensure the security and efficiency of Ethereum-based applications. 41](#_Toc171688595)

[4.4.9 Lecture 8 41](#_Toc171688596)

[Slide 117-118 41](#_Toc171688597)

[Etherscan IO 41](#_Toc171688598)

[Etherscan.io is a blockchain explorer for Ethereum, allowing users to search and view transactions, addresses, and smart contracts. It provides detailed information on the Ethereum blockchain, helping users track and verify activities on the network. 41](#_Toc171688599)

[4.5 Laboratory Sessions 42](#_Toc171688600)

[4.6 Private Study 49](#_Toc171688601)

[Topic 5: Private Blockchain 51](#_Toc171688602)

[5.1 Learning Objectives 51](#_Toc171688603)

[5.2 Pedagogic Approach 51](#_Toc171688604)

[5.3 Timings 51](#_Toc171688605)

[5.4 Lecture Notes 52](#_Toc171688606)

[5.4.1 Guidance on the Use of the Slides 52](#_Toc171688607)

[5.4.2 Lecture 1 52](#_Toc171688608)

[Slide 120-123 52](#_Toc171688609)

[What is Private Blockchain 52](#_Toc171688610)

[A private blockchain is a restricted-access network where participation is controlled by a single organization or a group of organizations. Unlike public blockchains, private blockchains offer enhanced privacy, faster transaction speeds, and more control over data and participants. 52](#_Toc171688611)

[5.4.3 Lecture 2 52](#_Toc171688612)

[Slide 124-127 52](#_Toc171688613)

[Smart Contract in Private Blockchain 52](#_Toc171688614)

[Smart contracts in private blockchains operate similarly to those in public blockchains but within a controlled environment. They automate and enforce business rules and agreements, ensuring transparency and efficiency in enterprise processes without compromising privacy. 52](#_Toc171688615)

[5.4.4 Lecture 3 52](#_Toc171688616)

[Slide 128-129 52](#_Toc171688617)

[State Machine in Smart Contract 52](#_Toc171688618)

[A state machine in smart contracts refers to a system that transitions between states based on predefined rules and events. Each state represents a particular condition of the contract, and transitions occur as specific conditions are met, enabling precise control over contract behavior. 52](#_Toc171688619)

[5.4.5 Lecture 4 53](#_Toc171688620)

[Slide 130-145 53](#_Toc171688621)

[Consensus Algorithms for Private Blockchain 53](#_Toc171688622)

[Consensus algorithms in private blockchains are designed to achieve agreement among trusted nodes with greater efficiency and speed. Common algorithms include Practical Byzantine Fault Tolerance (PBFT), which ensures consistency and fault tolerance, and Raft, which focuses on leader-based consensus for simplicity and performance. 53](#_Toc171688623)

[5.4.6 Lecture 5 53](#_Toc171688624)

[Slide 146-148 53](#_Toc171688625)

[Hyperledger Framework 53](#_Toc171688626)

[The Hyperledger framework is a collaborative open-source project hosted by the Linux Foundation, aiming to advance cross-industry blockchain technologies. It provides various tools and frameworks for developing enterprise-grade blockchain applications, emphasizing modularity and interoperability. 53](#_Toc171688627)

[5.4.7 Lecture 6 53](#_Toc171688628)

[Slide 149-151 53](#_Toc171688629)

[Hyperledger Tools 53](#_Toc171688630)

[Hyperledger offers several tools to facilitate blockchain development, including Hyperledger Composer for modeling business networks, Hyperledger Caliper for performance benchmarking, and Hyperledger Explorer for visualizing blockchain data. 53](#_Toc171688631)

[5.4.8 Lecture 7 53](#_Toc171688632)

[Slide 152-157 53](#_Toc171688633)

[Hyperledger Fabric Architecture 53](#_Toc171688634)

[Hyperledger Fabric is a modular and extensible blockchain platform designed for enterprise use. Its architecture includes key components like Membership Service Providers (MSPs) for identity management, chaincode for smart contracts, and a consensus mechanism that supports pluggable implementations for flexibility and scalability. 53](#_Toc171688635)

[5.4.9 Lecture 8 53](#_Toc171688636)

[Slide 158-162 53](#_Toc171688637)

[Supply Chain Management using Hyperledger 53](#_Toc171688638)

[Supply chain management using Hyperledger involves leveraging blockchain technology to enhance transparency, traceability, and efficiency in tracking goods and materials. By using Hyperledger Fabric, companies can create a tamper-proof record of transactions, streamline processes, and reduce fraud and errors in the supply chain. 53](#_Toc171688639)

[5.5 Laboratory Sessions 54](#_Toc171688640)

[5.6 Private Study 55](#_Toc171688641)

[Topic 6: Tools and Applications of Blockchain 57](#_Toc171688642)

[6.1 Learning Objectives 57](#_Toc171688643)

[6.2 Pedagogic Approach 57](#_Toc171688644)

[6.3 Timings 57](#_Toc171688645)

[6.4 Lecture Notes 58](#_Toc171688646)

[6.4.1 Guidance on the Use of the Slides 58](#_Toc171688647)

[6.4.2 Lecture 1 58](#_Toc171688648)

[Slide 164-166 58](#_Toc171688649)

[Hyperledger Fabric 58](#_Toc171688650)

[Hyperledger Fabric is a modular and permissioned blockchain framework designed for enterprise use. It supports plug-and-play components, including consensus and membership services, and allows for private and confidential transactions. Hyperledger Fabric is known for its flexibility, scalability, and ability to handle a wide range of business applications. 58](#_Toc171688651)

[6.4.3 Lecture 2 58](#_Toc171688652)

[Slide 167-170 58](#_Toc171688653)

[CORDA and Ripple 58](#_Toc171688654)

[Corda is a blockchain platform specifically designed for businesses to manage complex transactions securely and efficiently. It focuses on privacy and allows for direct transactions between parties. Ripple, on the other hand, is a real-time gross settlement system and remittance network aimed at enabling instant, cross-border payments. Ripple's consensus algorithm is designed to facilitate fast and low-cost international money transfers. 58](#_Toc171688655)

[6.4.4 Lecture 3 58](#_Toc171688656)

[Slide 171-176 58](#_Toc171688657)

[Quorum and DeFi 58](#_Toc171688658)

[Quorum is an enterprise-focused version of Ethereum that provides high-speed and high-throughput processing of private transactions within a permissioned network. It is optimized for use cases requiring data privacy and high performance. Decentralized Finance (DeFi) refers to financial systems built on public blockchains like Ethereum, allowing for decentralized lending, borrowing, and trading without traditional intermediaries. DeFi leverages smart contracts to provide transparent and trustless financial services accessible to anyone with an internet connection. 58](#_Toc171688659)

[6.5 Laboratory Sessions 59](#_Toc171688660)

[6.6 Private Study 62](#_Toc171688661)



Overview

# Module Overview and Objectives

In this blockchain course, students will begin by gaining a comprehensive understanding of fundamental blockchain concepts, including the structure and components of blockchain technology. They will learn to apply cryptographic hash functions, which are essential for ensuring the security and integrity of blockchain transactions. The course will also cover the development and deployment of smart contracts, enabling students to create decentralized applications with real-world utility. Additionally, students will design a public blockchain using Ethereum, exploring its architecture and capabilities, as well as a private blockchain using Hyperledger, understanding its specific use cases and advantages. Throughout the course, various blockchain tools will be utilized to provide hands-on experience in building and managing blockchain applications, preparing students for advanced roles in the evolving field of blockchain technology.

# Learning Outcomes and Assessment Criteria

|  |  |
| --- | --- |
| **Learning Outcomes;**  The Learner will: | **Assessment Criteria;**  The Learner can: |
| 1. Explain blockchain concepts | * 1. Describe the history and evolution of blockchain technology.   2. Identify and explain the key components of a blockchain, including blocks, chains, and nodes.   3. Differentiate between public, private, and consortium blockchains, and provide examples of each.   4. Explain the concept of decentralization and its importance in blockchain technology.   5. Illustrate the basic functioning of consensus mechanisms like Proof of Work (PoW) and Proof of Stake (PoS).   6. Discuss real-world applications of blockchain technology across various industries. |
| 2. Apply cryptographic hash required for blockchain. | * 1. Define cryptographic hash functions and explain their properties.   2. Demonstrate how cryptographic hash functions are used in blockchain for data integrity.   3. Explain the concept of hash pointers and their role in linking blocks within a blockchain.   4. Compare different cryptographic hash algorithms (e.g., SHA-256, Keccak-256) used in blockchain.   5. Implement a basic cryptographic hash function in a programming language (e.g., Python, Solidity).   6. Analyze the impact of hash collisions and how they are mitigated in blockchain systems. |
| 3. Apply the concepts of smart contracts for an application. | * 1. Define smart contracts and explain their significance in blockchain technology.   2. Identify different use cases for smart contracts across various industries.   3. Develop a simple smart contract using Solidity, covering basic structures and functions.   4. Test and deploy a smart contract on a testnet (e.g., Ropsten, Rinkeby) using tools like Remix IDE and Metamask.   5. Explain common vulnerabilities in smart contracts and methods to mitigate them.   6. Evaluate the performance and scalability of smart contracts in real-world applications. |
| 4. Design a public blockchain using Ethereum. | * 1. Describe the architecture and components of the Ethereum blockchain.   2. Set up a local Ethereum development environment using tools like Truffle and Ganache.   3. Develop and deploy a decentralized application (DApp) on the Ethereum blockchain.   4. Explain the role of Ether and gas in the Ethereum network.   5. Utilize Ethereum's native programming language, Solidity, to create and interact with smart contracts.   6. Analyze the transaction lifecycle in Ethereum and how it is recorded on the blockchain. |
| 5. Design a private blockchain using Hyperledger. | 1. Describe the architecture and key components of Hyperledger Fabric. 2. Set up a Hyperledger Fabric network using Docker and Compose. 3. Develop and deploy chaincode (smart contracts) on the Hyperledger Fabric network. 4. Explain the consensus mechanisms used in Hyperledger and their differences from public blockchains. 5. Evaluate the performance and scalability of Hyperledger Fabric for enterprise applications. 6. Demonstrate the integration of Hyperledger Fabric with other enterprise systems. |
| 6. Use different types of tools for blockchain applications. | 1. Identify and describe various blockchain development tools and frameworks (e.g., Truffle, Ganache, Remix IDE). 2. Set up and configure development environments for Ethereum and Hyperledger projects. 3. Utilize blockchain explorers (e.g., Etherscan, Blockchair) to monitor and analyze blockchain transactions. 4. Implement a project using an integrated development environment (IDE) suitable for blockchain development. 5. Use Metamask for managing Ethereum accounts and conducting transactions. 6. Evaluate the strengths and limitations of different blockchain development tools and choose the appropriate tool for specific tasks. |

# Syllabus

|  |  |  |  |
| --- | --- | --- | --- |
| Syllabus | | | |
| Topic No | Title | Proportion | Content |
| 1 | Introduction | 1/6  6 hours of lectures | * What is a blockchain, Origin of blockchain (cryptographically secure hash functions), Foundation of blockchain: Merkle trees * Components of blockchain, Block in blockchain, Types: Public, Private, and Consortium, Consensus Protocol, Limitations and Challenges of blockchain   ***Learning Outcome: 1*** |
| 2 | Cryptocurrency | 1/6 | * Bitcoin, Altcoin, and Tokens (Utility and Security), Cryptocurrency wallets: Hot and cold wallets, Cryptocurrency usage, Transactions in Blockchain, UTXO and double spending problem * Bitcoin blockchain: Consensus in Bitcoin, Proof-of-Work (PoW), Proof-of-Burn (PoB), Proof-of-Stake (PoS), and Proof-of-Elapsed Time (PoET), Life of a miner, Mining difficulty, Mining pool and its methods |
|  |  | 6 hours of lectures  1 hour of laboratory sessions | ***Learning Outcome: 2*** |
| 3 | Programming for Blockchain | 1/6  8 hours of lectures  1 hour of laboratory sessions | * Introduction to Smart Contracts, Types of Smart Contracts, Structure of a Smart Contract, Smart Contract Approaches, Limitations of Smart Contracts * Introduction to Programming: Solidity Programming – Basics, functions, Visibility and Activity Qualifiers, Address and Address Payable, Bytes and Enums, Arrays-Fixed and Dynamic Arrays, Special Arrays-Bytes and strings, Struct, Mapping, Inheritance, Error handling   ***Learning Outcome: 3*** |
| 4 | Public Blockchain | 1/6 | * Introduction to Smart Contracts, Types of Smart Contracts, Structure of a Smart Contract, Smart Contract Approaches, Limitations of Smart Contracts * Introduction to Programming: Solidity Programming – Basics, functions, Visibility and Activity Qualifiers, Address and Address Payable, Bytes and Enums, Arrays-Fixed and Dynamic Arrays, Special Arrays-Bytes and strings, Struct, Mapping, Inheritance, Error handling   ***Learning Outcome: 4*** |
|  |  | 8 hours of lectures |
|  |  | 2 hours of laboratory sessions |
| 5 | Private Blockchain | 1/6 | * Introduction, Key characteristics, Need of Private Blockchain, Smart Contract in a Private Environment, State Machine Replication, Consensus Algorithms for Private Blockchain - PAXOS and RAFT, Byzantine Faults: Byzantine Fault Tolerant (BFT) and Practical BFT * Introduction to Hyperledger, Tools and Frameworks, Hyperledger Fabric, Comparison between Hyperledger Fabric & Other Technologies * Hyperledger Fabric Architecture, Components of Hyperledger Fabric: MSP, Chain Codes, Transaction Flow, Working of Hyperledger Fabric, Creating Hyperledger Network, Case Study of Supply Chain Management using Hyperledger   ***Learning Outcome: 5*** |
|  |  | 8 hours of lectures |
|  |  | 2 hours of laboratory sessions |
| 6 | Tools and Applications of Blockchain | 1/6 | * Corda, Ripple, Quorum and other Emerging Blockchain Platforms, Blockchain in DeFi: Case Study on any of the Blockchain Platforms.   ***Learning Outcome: 6*** |
|  |  | 3 hours of |
|  |  | lectures |
|  |  | 3 hours of |
|  |  | laboratory |
|  |  | sessions |

# Related National Occupational Standards

The UK National Occupational Standards describe the skills that professionals are expected to demonstrate in their jobs in order to carry them out effectively. They are developed by employers and this information can be helpful in explaining the practical skills you have covered in this module.

**Sector Subject Area: Related NOS:**

**Related National Occupational Standards (NOS)**

# Resources

Lecturer Guide: This guide contains notes for lecturers on the organisation of each topic, and suggested use of the resources. It also contains all of the suggested exercises and model answers.

PowerPoint Slides: These are presented for each topic for use in the lectures. They contain many examples which can be used to explain the key concepts. Handout versions of the slides are also available; it is recommended that these are distributed to students for revision purposes as it is important that students learn to take their own notes during lectures.

Student Guide: This contains the topic overviews and all of the suggested exercises. Students should bring this guide to every lecture and laboratory session throughout the module.

Software: Remix IDE: An online integrated development environment for writing, testing, and deploying smart contracts using Solidity.

Visual Studio Code: A powerful code editor with various extensions for blockchain development (e.g., Solidity, Hyperledger).

Truffle: A development environment, testing framework, and asset pipeline for Ethereum.

Ganache: A personal blockchain for Ethereum development to deploy contracts, develop applications, and run tests.

Metamask: A browser extension and mobile app that allows users to interact with the Ethereum blockchain, manage accounts, and sign transactions.

# Pedagogic Approach

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Suggested Learning Hours** | | | | | | |
| **Guided Learning Hours** | | | | **Assessment** | **Private Study:** | **Total:** |
| **Lectures:** | **Tutorial:** | **Seminar:** | **Laboratory:** |
| 39 | - | - | 18 | 30 hours (assignment) | 60 hours | 150 |

The teacher-led time for this module is comprised of lectures and laboratory sessions. The breakdown of the hours is also given at the start of each topic.

## Lectures

Lectures are designed to start each topic and PowerPoint slides are presented for use during these sessions. Students should also be encouraged to be active during this time and to discuss the concepts covered. Lecturers should encourage active participation wherever possible. The use of question-and-answer sessions within lectures is particularly recommended to develop the concepts in relation to the networks students are familiar with.

## Laboratory Sessions

Laboratory sessions are designed to follow the respective topic lecture. The purpose is to develop concepts in relation to real life networks and provide opportunity to develop the skills required to meet the practical assessment criteria. During these sessions, students should work through practical tutorials and various exercises. The details of these are provided in this guide and also in the Student Guide.

## Private Study

In addition to the taught portion of the module, students will also be expected to undertake private study. Suggested exercises are provided in the Student Guide for students to complete during this time. Teachers will need to set deadlines for the completion of this work.

# Assessment

This module will be assessed by means of an assignment worth 100% of the total mark. The assessment will be based on the assessment criteria given above and students will be expected to demonstrate that they have met the module’s learning outcomes.

# Further Reading List

A selection of sources of further reading around the content of this module must be available in your Accredited Partner Centre’s library. The following list provides suggestions of some suitable sources:

Blockchain Technology, Chandramouli Subramanian, Asha A. George, Abhillash K. A and Meena Karthikeyen, Universities Press

--

Mastering Ethereum, Building Smart Contract and Dapps Andreas M. Antonopoulos Dr. Gavin Wood O‘reilly.

--

Mastering Blockchain: A deep dive into distributed ledgers, consensus protocols, smart contracts, DApps, cryptocurrencies, Ethereum, and more,

Imran Bashir, Packt Publishing 3rd Edition,



Topic 1

# Topic 1: Introduction

## Learning Objectives

Blockchain technology, a revolutionary innovation, is fundamentally a decentralized and immutable ledger system that records transactions across multiple computers. It ensures transparency, security, and integrity, transforming various industries by enabling trustless and efficient transactions.

On completion of the topic, students will be able to:

* Identify the fundamental concepts and components of blockchain technology. (Knowledge)
* Explain the origin and development of blockchain and its key components. (Comprehension)
* Differentiate between Public, Private, and Consortium blockchains. (Application)
* Describe various consensus protocols and their importance. (Analysis)
* Identify the limitations and challenges of blockchain technology. (Evaluation)
* Analyze real-world applications of blockchain in various industries. (Synthesis)

## Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then investigate the topic further during the laboratory sessions. Private study exercises will be provided to further enhance the students’ understanding of the topic.

## Timings

Lectures: 6 hours Laboratory Sessions: 0 hour Private Study: 6 hours

## Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

* Definition of Blockchain
* How blockchain technology works
* Components of Blockchain
* Types of Blockchain
* Consensus Algorithm
* Pros and Cons of Blockchain

### Guidance on the Use of the Slides

The slides are divided into four lectures, each lasting 1 hour. These may be delivered as six separate lectures or you may combine them into longer sessions.

### Lecture 1

|  |  |
| --- | --- |
| Slide 4-5: | Definition of Blockchain: Blockchain is a distributed, peer-to-peer network that allows multiple non-trusting parties to transact without a trusted intermediary, maintaining an ever-growing list of time-sequenced records. |

### Lecture 2

|  |  |
| --- | --- |
| Slide 6: | How BLOCKCHAIN technology works: Blockchain works by initiating and broadcasting transactions to a network of nodes that validate them. Verified transactions are combined into a block, which is then added to the blockchain by miners using consensus algorithms, making the transaction immutable. |

### Lecture 3

|  |  |
| --- | --- |
| Slide 7-20: | Components of BLOCKCHAIN: Blockchain components include nodes (devices connected to the network), a ledger (an immutable, distributed database), wallets (digital tools for managing cryptocurrencies), nonce (a random number for block creation), hash (a unique identifier for data), and mining (the process of validating transactions and adding them to the ledger). |

### Lecture 4

|  |  |
| --- | --- |
| Slide 21-27 | Types of Blockchain: There are three types of blockchains: public (accessible to anyone), private (restricted access), and consortium (controlled by a group of organizations). |

### Lecture 5

|  |  |
| --- | --- |
| Slide 28-32 | Consensus Algorithm: A consensus algorithm is a procedure used in blockchain networks to achieve agreement on the present state of the data among distributed nodes. This ensures that all participants in the network have the same data. Common consensus algorithms include Proof of Work (PoW), where miners solve complex puzzles to add a block, and Proof of Stake (PoS), where validators are chosen based on the number of coins they hold and are willing to "stake" as collateral. |

### Lecture 5

|  |  |
| --- | --- |
| Slide 33-25 | Pros and Cons of Blockchain: Blockchain offers decentralization and transparency by eliminating the need for a central authority and distributing data across multiple nodes, ensuring greater security and trust. Additionally, its immutability feature guarantees that once data is recorded, it cannot be altered or deleted, providing a permanent and tamper-proof record. However, blockchain faces scalability issues, often being slower and less efficient than traditional databases, especially as the size of the blockchain grows. Furthermore, certain consensus algorithms, such as Proof of Work, require significant computational power, resulting in high energy consumption. |

## Laboratory Sessions

The laboratory time allocation for this topic is 0 hour.

**Lecturers’ Notes:**

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide. They should work in small groups to complete the exercise below.

## Private Study

The time allocation for private study in this topic is expected to be approximately 5 hours.

**Lecturers’ Notes:**

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide. They are expected to also use private study time to review the content of this unit.

You may wish to ask students to complete the private study exercises after relevant lecture has been delivered and then allow time for review of their answers in the next lecture or laboratory session. Alternatively, you may prefer to review the answers during Topic 2 or to collect in written work for one or more exercises as you feel is appropriate.

#### Exercise 1:

Research the development of blockchain technology from its inception to the present day. Provide a brief overview of its evolution through the decades, focusing on key milestones and advancements.

#### Suggested Answer:

There are many areas a student could concentrate on, such as the technological advancements, the introduction of cryptocurrencies, and the adoption of blockchain in various industries. All approaches are suitable for this task; the aim is to allow the student to develop an understanding of the major milestones in blockchain's history. The answers given below are merely suggestive of some topics which students may raise.

2000s: Early theoretical work on cryptographically secure hash functions and decentralized systems. In 2008, Satoshi Nakamoto publishes the Bitcoin white paper, proposing a peer-to-peer electronic cash system. In 2009, the first block of the Bitcoin blockchain, known as the Genesis Block, is mined.

2010s: Bitcoin gains popularity, and other cryptocurrencies like Ethereum are introduced, expanding the capabilities of blockchain to include smart contracts. The concept of blockchain as a decentralized, immutable ledger gains traction beyond cryptocurrencies, with applications in finance, supply chain, and healthcare emerging. Enterprises begin exploring private and consortium blockchains.

2020s: Continued innovation in blockchain technology, including advancements in scalability, interoperability, and energy-efficient consensus mechanisms. Increased adoption of blockchain in various sectors such as DeFi (Decentralized Finance), NFTs (Non-Fungible Tokens), and supply chain management. Governments and regulatory bodies start developing frameworks to govern blockchain and cryptocurrency use.

#### Exercise 2:

Research one significant impact that blockchain technology has had on financial transactions. Explain what the impact is and describe how blockchain technology has brought about this change. Discuss the advantages and drawbacks of implementing blockchain in financial transactions, using details from news articles or journal reports to highlight relevant cases in the public spotlight.

You should write 500-1000 words.

#### Suggested Answer:

Students should concentrate on one significant impact discussed in lectures, such as the decentralization of financial systems, the reduction of transaction costs, or the increase in transaction speed. They should also include details of the benefits and potential drawbacks of blockchain technology in financial transactions. More able students should be able to synthesize a view on the overall benefits and challenges of using blockchain in financial systems.

For example:

Impact: Blockchain has enabled decentralized finance (DeFi), allowing users to conduct financial transactions without intermediaries like banks.

Description: DeFi platforms leverage blockchain to offer services such as lending, borrowing, and trading directly between participants. This reduces costs, increases transaction speeds, and provides financial services to the unbanked population.

Advantages: Lower transaction fees, increased security and transparency, and broader access to financial services.

Drawbacks: Regulatory uncertainty, potential security vulnerabilities, and the high energy consumption of certain consensus mechanisms.

Case Study: The rise of platforms like Uniswap and Compound, which have gained significant traction and highlighted both the potential and challenges of DeFi.



Topic 2

# Topic 2: Cryptocurrency

## Learning Objectives

Cryptocurrency represents a revolutionary digital asset designed to work as a medium of exchange using cryptography to secure transactions. Originating with Bitcoin, the first decentralized cryptocurrency introduced in 2009, this chapter explores the evolution of cryptocurrencies, their underlying blockchain technology, and their impact on global financial systems. It also examines various types of cryptocurrencies, wallets, and the mechanisms ensuring secure and transparent transactions.

On completion of the topic, students will be able to:

* Define key concepts related to cryptocurrencies, including Bitcoin, Altcoins, and Tokens. (Knowledge)
* Explain the functionality and types of cryptocurrency wallets. (Comprehension)
* Describe the processes involved in cryptocurrency transactions. (Application)
* Analyze different consensus mechanisms in Bitcoin, such as PoW and PoS. (Analysis)
* Discuss the life of a miner and mining pool methods. (Evaluation)
* Evaluate security concerns and solutions related to double spending. (Synthesis)

## Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then investigate the topic further during the laboratory sessions. Private study exercises will be provided to further enhance the students’ understanding of the topic.

## Timings

Lectures: 6 hours Laboratory Sessions: 0 hour Private Study: 6 hours

## Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

* Transaction in Blockchain
* Double Spending Problem
* UTXO
* Cryptocurrency Wallets
* Altcoins and Tokens
* Mining protocols and terminologies

### Guidance on the Use of the Slides

The slides are divided into four lectures, each lasting 1 hour. These may be delivered as four separate lectures or you may combine them into longer sessions.

### Lecture 1

|  |  |
| --- | --- |
| Slide 37 | Transaction in Blockchain: Transactions in blockchain involve a process where a node requests a transaction, which is then broadcasted to the network. The transaction is validated by the network using consensus algorithms and added to a block, making it permanent and immutable. |

### Lecture 2

|  |  |
| --- | --- |
| Slide 38-39 | Double Spending Problem: The double spending problem occurs when the same digital currency is spent more than once. Blockchain solves this issue using consensus mechanisms that make it nearly impossible to duplicate a transaction once it is confirmed. |

### Lecture 3

|  |  |
| --- | --- |
| Slide 40-41 | UTXO: UTXO refers to the amount of digital currency remaining after a cryptocurrency transaction is executed. These unspent outputs are used as inputs for future transactions, ensuring accurate and traceable accounting of funds. |

### Lecture 4

|  |  |
| --- | --- |
| Slide 42-47 | Cryptocurrency Wallets: Cryptocurrency wallets are software programs that store private and public keys, allowing users to send, receive, and manage their digital assets. They come in various forms, including hot wallets (online), cold wallets (offline), hardware wallets, and paper wallets. |

### Lecture 5

|  |  |
| --- | --- |
| Slide 48-51 | Altcoins: Altcoins are alternative cryptocurrencies to Bitcoin, created by forking the Bitcoin or other blockchain code. They often offer different features, improved functionality, or enhanced security compared to their predecessors.Tokens: Tokens are digital assets created on existing blockchains, representing either a utility (access to a service) or security (investment). They are commonly used in fundraising through Initial Coin Offerings (ICOs) and for transactions within decentralized applications. |

### Lecture 6

|  |  |
| --- | --- |
| Slide 52-67 | Mining Protocols  Mining protocols are the rules and methods used to validate transactions and add them to the blockchain. Various consensus algorithms ensure that all participants in the network agree on the validity of transactions. Key protocols include Proof of Work (PoW), where miners compete to solve complex puzzles; Proof of Stake (PoS), which selects validators based on the number of coins they hold; and Proof of Burn, where coins are destroyed to earn the right to mine new blocks .  Mining Terminologies  Hash: A unique identifier generated by processing data through a cryptographic algorithm. It is akin to a fingerprint for data.  Nonce: A random number used once in cryptographic communication, crucial for mining as it ensures each transaction's uniqueness.  Mining Difficulty: Refers to the complexity of the puzzles that miners must solve to create a new block, which adjusts periodically to maintain a steady rate of block creation.  Mining Pool: A group of miners who combine their resources to increase the chances of successfully mining a block and sharing the rewards. Block Frequency: The regular interval at which new blocks are added to the blockchain, such as every ten minutes in Bitcoin . |

## Laboratory Sessions

The laboratory time allocation for this topic is 0 hour.

**Lecturers’ Notes:**

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide. You may wish to organise students into small groups to conduct the exercise below.

## Private Study

The time allocation for private study in this topic is expected to be 6 hours.

**Lecturers’ Notes:**

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide. They are expected to also use private study time to review the content of this unit.

You may wish to ask students to complete the private study exercises after the relevant lecture has been delivered and then allow time for review of their answers in the next lecture or to collect in written work for one or more exercises as you feel is appropriate.

#### Exercise 1:

#### Research and analyze the process of a blockchain transaction from initiation to completion. Describe each step in detail and explain the role of nodes, consensus algorithms, and blocks in ensuring the transaction's security and immutability.

#### Suggested Answer:

A thorough answer should include:

Initiation: How a transaction is requested and broadcasted to the network.

Validation: The role of nodes in validating the transaction using consensus algorithms.

Block Creation: How validated transactions are grouped into blocks.

Consensus: The process of reaching consensus to add the block to the blockchain.

Finalization: How the transaction is confirmed and becomes immutable.

#### Exercise 2:

Investigate the double spending problem in digital currencies and how blockchain technology addresses this issue. Provide examples of potential double spending scenarios and discuss the mechanisms that blockchain uses to prevent them.

#### Suggested Answer:

#### A comprehensive answer should cover:

#### Definition: Explain what double spending is and why it poses a problem in digital currencies.

#### Scenarios: Provide examples of how double spending could occur without blockchain.

#### Blockchain Solution: Describe how consensus mechanisms like Proof of Work and Proof of Stake prevent double spending.

#### Case Study: Reference a specific instance or study where blockchain effectively prevented double spending.

#### Exercise 3:

Research and write a detailed report on the different types of cryptocurrency wallets available. Include an explanation of each type, their features, security aspects, and use cases. Compare and contrast the advantages and disadvantages of hot wallets, cold wallets, hardware wallets, and paper wallets. Provide examples of popular wallets in each category and discuss how users can choose the most appropriate wallet based on their needs.

#### Suggested Answer:

#### Introduction

#### Cryptocurrency wallets are essential tools for managing digital assets. They store private and public keys and enable users to send, receive, and track their cryptocurrency holdings. There are various types of wallets, each with unique features and security considerations.

#### Types of Cryptocurrency Wallets

#### Hot Wallets

#### Definition: Hot wallets are online wallets connected to the internet, making them accessible from anywhere.

#### Features: Easy to set up and use, often free, suitable for frequent transactions.

#### Security: Vulnerable to hacking due to constant internet connection.

#### Use Cases: Ideal for daily transactions and small amounts of cryptocurrency.

#### Examples: MetaMask, MyEtherWallet.

#### Cold Wallets

#### Definition: Cold wallets are offline wallets, not connected to the internet, providing enhanced security.

#### Features: Highly secure, used for long-term storage.

#### Security: Immune to online hacking attempts.

#### Use Cases: Suitable for storing large amounts of cryptocurrency for extended periods.

#### Examples: Paper wallets, hardware wallets.

#### Hardware Wallets

#### Definition: Physical devices that securely store private keys offline.

#### Features: Portable, can be connected to a computer when needed, provides a high level of security.

#### Security: Protected from online threats, but must be kept physically secure.

#### Use Cases: Best for users who need both security and the ability to make transactions occasionally.

#### Examples: Ledger Nano S, Trezor.

#### Paper Wallets

#### Definition: Physical pieces of paper with printed public and private keys, often in the form of QR codes.

#### Features: Simple and secure if stored properly, completely offline.

#### Security: Highly secure from online threats, but vulnerable to physical damage and loss.

#### Use Cases: Suitable for long-term storage by users who can ensure the physical security of the paper wallet.

#### Examples: Generated using services like BitAddress.

#### Comparison of Wallet Types

#### Hot Wallets:

#### Advantages: Convenience, ease of use, free or low cost.

#### Disadvantages: High vulnerability to cyber-attacks, not suitable for large amounts.

#### Cold Wallets:

#### Advantages: Enhanced security, ideal for long-term storage.

#### Disadvantages: Less convenient for frequent transactions, requires physical security measures.

#### Hardware Wallets:

#### Advantages: High security, portable, easy to use for transactions when needed.

#### Disadvantages: Can be expensive, requires careful handling to avoid physical loss or damage.

#### Paper Wallets:

#### Advantages: Extremely secure from online threats, no cost to create.

#### Disadvantages: Prone to physical damage, loss, and deterioration over time.

#### Choosing the Right Wallet

#### Users should consider their specific needs when choosing a cryptocurrency wallet. Factors to consider include the amount of cryptocurrency they hold, how often they need to make transactions, and their comfort level with managing digital security. For daily transactions and small amounts, hot wallets are suitable. For long-term storage of large amounts, cold wallets, especially hardware wallets, are recommended. Users who prioritize security and can manage physical storage may opt for paper wallets.

#### Conclusion

#### Cryptocurrency wallets are crucial for managing digital assets securely. Understanding the differences between hot wallets, cold wallets, hardware wallets, and paper wallets allows users to make informed decisions based on their needs. By carefully considering the advantages and disadvantages of each type, users can ensure the safety and accessibility of their cryptocurrency holdings.

#### References: Students should support their reports with references from credible sources, such as articles, journals, and official wallet websites.



Topic 3

# Topic 3: Programming in Blockchain

## Learning Objectives

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They run on blockchain platforms, enabling automated, secure, and transparent transactions without the need for intermediaries. This chapter explores the structure, types, and applications of smart contracts, along with the programming languages used to create them.

On completion of the topic, students will be able to:

* Define what a smart contract is and its purpose in blockchain. (Knowledge)
* Differentiate between types of smart contracts and their applications. (Comprehension)
* Develop and deploy basic smart contracts using Solidity. (Application)
* Analyze the structure and components of a smart contract. (Analysis)
* Identify the limitations and challenges of smart contracts. (Evaluation)
* Apply error handling and inheritance concepts in Solidity programming. (Synthesis)

## Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then investigate the topic further during the laboratory sessions. Private study exercises will be provided to further enhance the students’ understanding of the topic.

## Timings

Lectures: 8 hours Laboratory Sessions : 8 hour Private Study: 8 hours

## Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

* Introduction to smart contract
* Components of smart contract
* Smart contract approaches
* Data types in solidity
* Functions in solidity
* Visibility Quantifiers
* Inheritance
* Error handling

### Guidance on the Use of the Slides

The slides are divided into four lectures, each lasting 1 hour. These may be delivered as eight separate lectures or you may combine them into longer sessions.

### Lecture 1

|  |  |
| --- | --- |
| Slide 69-75 | Introduction to smart contract Smart contracts are self-executing digital contracts with the terms of the agreement directly written into lines of code. These contracts automatically enforce and execute the agreement once predefined conditions are met, ensuring trust and transparency without the need for intermediaries. |

### Lecture 2

|  |  |
| --- | --- |
| Slide 76-80 | Components of Smart ContractThe main components of a smart contract include variables (to store data), functions (to define the contract's operations), and events (to log activities). These elements work together to manage the contract's state and interactions on the blockchain. |

### Lecture 3

|  |  |
| --- | --- |
| Slide 81-83 | Smart Contract ApproachesSmart contract approaches involve different methods to create and deploy contracts, including using specific programming languages like Solidity, defining clear terms and conditions, and ensuring security measures to prevent vulnerabilities and attacks. |

### Lecture 4

|  |  |
| --- | --- |
| Slide 84-88 | Data Types in SoliditySolidity, the primary language for Ethereum smart contracts, supports various data types including integers, booleans, strings, and arrays. These data types are used to store and manipulate information within the contract. |

### Lecture 5

|  |  |
| --- | --- |
| Slide 89-91 | Functions in SolidityFunctions in Solidity define the behavior of a smart contract. They can be public or private, allowing interaction with the contract's data. Functions can also be marked as "view" or "pure" to indicate they do not modify the contract's state. |

### Lecture 6

|  |  |
| --- | --- |
| Slide 92 | Visibility QuantifiersVisibility quantifiers in Solidity determine who can call a function or access a variable. The main visibility types are public, private, internal, and external, each providing different levels of access control. |

### Lecture 7

|  |  |
| --- | --- |
| Slide 93 | InheritanceInheritance in Solidity allows one contract to inherit properties and functions from another, promoting code reusability and modularity. This enables developers to build complex systems by extending existing contracts. |

### Lecture 8

|  |  |
| --- | --- |
| Slide 94 | Error HandlingError handling in Solidity involves using mechanisms like require(), assert(), and revert() to ensure the contract functions as intended. These tools help manage errors and validate conditions, enhancing the contract's security and reliability. |

## Laboratory Sessions

The laboratory time allocation for this topic is 8 hours.

**Lecturers’ Notes:**

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

#### Exercise 1:

Introduction to Ethereum and Setting Up MetaMask

#### Suggested Answer:

Introduction to Ethereum:

Explain Ethereum as a decentralized platform that runs smart contracts.

Discuss the Ethereum blockchain, its native currency Ether (ETH), and the concept of gas for transaction fees.

Setting Up MetaMask:

Install MetaMask as a browser extension (available for Chrome, Firefox, etc.).

Create a new MetaMask wallet by following the on-screen instructions.

Securely store the seed phrase provided during wallet creation.

Add the MetaMask extension to the browser toolbar for easy access.

Verification:

After setting up, the MetaMask wallet should display a unique Ethereum address.

Ensure the wallet is connected to the Ethereum Mainnet or a testnet (like Ropsten or Rinkeby) for practice.

#### Exercise 2:

Write and deploy a basic smart contract on the Ethereum test network using Remix IDE.

#### Suggested Answer:

Introduction to Remix IDE:

Open the Remix IDE (https://remix.ethereum.org) in a web browser.

Create a new file in Remix and name it SimpleStorage.sol.

Writing the Smart Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract SimpleStorage {

uint256 public storedData;

function set(uint256 x) public {

storedData = x;

}

function get() public view returns (uint256) {

return storedData;

}

}

Deploying the Smart Contract:

Compile the contract using the Solidity compiler in Remix.

Select the "Deploy & Run Transactions" tab, choose "Injected Web3" as the environment, which connects to MetaMask.

Deploy the contract by clicking the "Deploy" button and confirm the transaction in MetaMask.

Verification:

After deployment, the deployed contract should appear in Remix under "Deployed Contracts".

Use the set and get functions to interact with the contract and verify its functionality.

#### Exercise 3:

Interact with a deployed smart contract to understand how to call its functions and retrieve data.

#### Suggested Answer:

Access Deployed Contract:

Open Remix IDE and navigate to the "Deployed Contracts" section.

Select the deployed SimpleStorage contract.

Interacting with Functions:

Use the set function to store a new value (e.g., 42). Confirm the transaction in MetaMask.

Call the get function to retrieve the stored value and verify that it returns 42.

Verification:

Ensure that the value set using the set function is correctly retrieved using the get function, confirming successful interaction.

#### Exercise 4:

Develop and deploy a basic ERC-20 token contract on the Ethereum test network.

#### Suggested Answer:

Writing the Token Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract MyToken is ERC20 {

constructor(uint256 initialSupply) ERC20("MyToken", "MTK") {

\_mint(msg.sender, initialSupply);

}

}

Deploying the Token Contract:

Compile the contract in Remix IDE.

Deploy the contract by specifying an initial supply (e.g., 1000000 tokens).

Verification:

Verify the token creation by checking the balance of the deployer address.

Use MetaMask to see the newly created token under the assets section.

## Private Study

The time allocation for private study in this topic is expected to be 8 hours.

**Lecturers’ Notes:**

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide. They are expected to also use private study time to review the content of this topic.

You may wish to ask students to complete the private study exercises after the relevant lecture has been delivered and then allow time for review of their answers in the next lecture or laboratory session. Alternatively, you may prefer to review the answers during Topic 4 or to collect in written work for one or more of the exercises as you feel is appropriate.

#### Exercise 1:

Gain a comprehensive understanding of different data types and variables in Solidity.

Research the various data types available in Solidity, including integers, booleans, strings, arrays, structs, and mappings.

Write a detailed report explaining each data type and provide examples of how they are used in smart contracts.

Create a simple Solidity contract that demonstrates the use of at least three different data types, showing how they can be declared, assigned values, and accessed.

#### Suggested Answer:

Introduction to Solidity data types

Explanation of primitive data types: integers, booleans, and strings

Explanation of complex data types: arrays, structs, and mappings

Example Solidity contract utilizing multiple data types

Example Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract DataTypesExample {

// Integer variable

uint256 public number;

// Boolean variable

bool public isTrue;

// String variable

string public text;

// Array variable

uint256[] public numbersArray;

// Struct variable

struct Person {

string name;

uint256 age;

}

Person public person;

// Mapping variable

mapping(address => uint256) public balances;

// Function to set values

function setValues(uint256 \_number, bool \_isTrue, string memory \_text, uint256 \_age) public {

number = \_number;

isTrue = \_isTrue;

text = \_text;

person = Person("Alice", \_age);

numbersArray.push(\_number);

balances[msg.sender] = \_number;

}

}

#### Exercise 2:

Learn about different types of functions in Solidity and their visibility quantifiers.

Task:

Research the different types of functions in Solidity, including public, private, internal, and external functions.

Write a detailed report explaining the use cases and differences between these function types.

Implement a Solidity contract that includes examples of each function type, demonstrating their visibility and accessibility within the contract and from external sources.

#### Suggested Answer:

Introduction to Solidity functions

Explanation of function visibility: public, private, internal, and external

Use cases for each function type

Example Solidity contract demonstrating function types

Example Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract FunctionsExample {

uint256 private value;

// Public function

function setValue(uint256 \_value) public {

value = \_value;

}

// Private function

function getValue() private view returns (uint256) {

return value;

}

// Internal function

function incrementValue() internal {

value += 1;

}

// External function

function callIncrement() external {

incrementValue();

}

// Public function to demonstrate visibility

function showValue() public view returns (uint256) {

return getValue();

}

}

#### Exercise 3:

Understand and implement error handling mechanisms in Solidity.

Task:

Research the error handling mechanisms available in Solidity, including require(), assert(), and revert().

Write a detailed report explaining the use cases and differences between these error handling functions.

Implement a Solidity contract that includes examples of each error handling mechanism, demonstrating how they can be used to ensure the contract functions correctly and securely.

#### Suggested Answer:

Introduction to error handling in Solidity

Explanation of require(), assert(), and revert() functions

Use cases and differences between the error handling functions

Example Solidity contract demonstrating error handling

Example Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract ErrorHandlingExample {

uint256 public value;

// Function using require()

function setValue(uint256 \_value) public {

require(\_value > 0, "Value must be greater than zero");

value = \_value;

}

// Function using assert()

function checkValue() public view {

assert(value > 0);

}

// Function using revert()

function resetValue() public {

if (value == 0) {

revert("Value is already zero");

}

value = 0;

}

}



Topic 4

# Topic 4: Public Blockchain

## Learning Objectives

Public blockchains are decentralized networks accessible to anyone, providing transparency and security without a central authority. This chapter focuses on Ethereum, one of the most prominent public blockchains, exploring its components, mining process, and the Ethereum Virtual Machine (EVM). It also covers the development of decentralized applications (DApps) and compares Ethereum with other public blockchains like Bitcoin.

On completion of the topic, students will be able to:

* Describe the components and architecture of Ethereum as a public blockchain. (Knowledge)
* Explain the process of mining in Ethereum and the role of the Ethereum Virtual Machine (EVM). (Comprehension)
* Compare and contrast Bitcoin and Ethereum blockchains. (Application)
* Use tools like Metamask and Mist Wallet for transferring Ethers and managing transactions. (Analysis)
* Analyze the structure of Ether blocks and explore etherscan.io. (Evaluation)
* Evaluate and propose different test-networks used in Ethereum for specific applications. (Creation)

## Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then investigate the topic further during the laboratory sessions. Private study exercises will be provided to further enhance the students’ understanding of the topic.

## Timings

Lectures: 8 hours Laboratory Sessions:10 hours Private Study: 8 hours

## Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

* Ethereum Components
* Mining in Ethereum
* Architecture of Ethereum
* Workflow of Ethereum
* Comparison between bitcoin and Ethereum
* Transferring ethers using metamask
* Ethereum Frameworks
* Etherscan IO

### Guidance on the Use of the Slides

The slides are divided into three lectures, each lasting 1 hour. These may be delivered as three separate lectures or you may combine them into longer sessions.

### Lecture 1

|  |  |
| --- | --- |
| Slide 96-100 | Ethereum ComponentsEthereum comprises several key components, including smart contracts, decentralized applications (DApps), Ether (ETH) as its native cryptocurrency, and the Ethereum Virtual Machine (EVM) which executes smart contracts. These elements work together to create a versatile and programmable blockchain platform. |

### Lecture 2

|  |  |
| --- | --- |
| Slide 101-105 | Mining in EthereumMining in Ethereum involves validating transactions and creating new blocks by solving complex cryptographic puzzles. Miners compete to solve these puzzles using a Proof of Work (PoW) consensus mechanism, which secures the network and adds verified blocks to the blockchain. |

### Lecture 3

|  |  |
| --- | --- |
| Slide 106-107 | Architecture of EthereumThe architecture of Ethereum includes nodes, the Ethereum Virtual Machine (EVM), smart contracts, and a decentralized network. Nodes maintain the blockchain, the EVM executes code, and smart contracts automate transactions, creating a robust and programmable blockchain ecosystem. |

### Lecture 4

|  |  |
| --- | --- |
| Slide 108-109 | Workflow of EthereumThe workflow of Ethereum involves creating and deploying smart contracts, processing transactions, and achieving consensus through mining. Transactions are initiated by users, validated by miners, and recorded on the blockchain, ensuring transparency and immutability. |

### Lecture 5

|  |  |
| --- | --- |
| Slide 110-111 | Comparison between Bitcoin and EthereumBitcoin and Ethereum are both prominent public blockchains but differ in purpose and functionality. Bitcoin focuses on secure, decentralized digital currency transactions, while Ethereum provides a platform for building and executing smart contracts and decentralized applications. |

### Lecture 6

|  |  |
| --- | --- |
| Slide 112-114 | Transferring Ethers Using MetaMaskMetaMask is a browser extension that allows users to manage their Ethereum wallets and interact with the Ethereum blockchain. Users can transfer Ether by connecting MetaMask to the network, entering the recipient's address, specifying the amount, and confirming the transaction. |

### Lecture 7

|  |  |
| --- | --- |
| Slide 115-116 | Ethereum FrameworksEthereum frameworks such as Truffle, Hardhat, and OpenZeppelin provide tools and libraries for developing, testing, and deploying smart contracts. These frameworks simplify the development process and ensure the security and efficiency of Ethereum-based applications. |

### Lecture 8

|  |  |
| --- | --- |
| Slide 117-118 | Etherscan IOEtherscan.io is a blockchain explorer for Ethereum, allowing users to search and view transactions, addresses, and smart contracts. It provides detailed information on the Ethereum blockchain, helping users track and verify activities on the network. |

## Laboratory Sessions

The laboratory time allocation for this topic is 10 hours.

**Lecturers’ Notes:**

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

#### Exercise 1:

Develop a user interface for a decentralized application (DApp) on Ethereum.

#### Suggested Answer:

Set Up Development Environment:

Install Node.js and npm.

Install Truffle framework (npm install -g truffle).

Initialize a new Truffle project (truffle init).

Write a Simple Smart Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract SimpleDApp {

string public message;

function setMessage(string memory newMessage) public {

message = newMessage;

}

}

Save this contract in the contracts directory (e.g., SimpleDApp.sol).

Deploy the Smart Contract:

Configure the Truffle project to use Ganache or a testnet.

Create a migration script in the migrations directory.

Deploy the contract using truffle migrate.

Build the DApp Interface:

Create a simple frontend using HTML, CSS, and JavaScript.

Use Web3.js to interact with the deployed smart contract.

Example interface code:

html

Copy code

<!DOCTYPE html>

<html>

<head>

<title>Simple DApp</title>

<script src="https://cdn.jsdelivr.net/npm/web3@1.3.0/dist/web3.min.js"></script>

</head>

<body>

<h1>Simple DApp</h1>

<input type="text" id="messageInput" placeholder="Enter message">

<button onclick="setMessage()">Set Message</button>

<p id="currentMessage">Current message: </p>

<script>

const contractAddress = 'YOUR\_CONTRACT\_ADDRESS';

const abi = [/\* ABI array \*/];

const web3 = new Web3(Web3.givenProvider);

const contract = new web3.eth.Contract(abi, contractAddress);

async function setMessage() {

const accounts = await web3.eth.requestAccounts();

const message = document.getElementById('messageInput').value;

await contract.methods.setMessage(message).send({ from: accounts[0] });

loadMessage();

}

async function loadMessage() {

const message = await contract.methods.message().call();

document.getElementById('currentMessage').innerText = `Current message: ${message}`;

}

loadMessage();

</script>

</body>

</html>

Verification:

Ensure the DApp interface allows setting and retrieving the message from the smart contract.

#### Exercise 2:

Analyze a real-world application of Ethereum, such as a DeFi application or supply chain management system.

#### Suggested Answer:

Select a Case Study:

Choose a specific real-world Ethereum application, such as Uniswap (DeFi) or IBM's Food Trust (supply chain).

Research and Analysis:

Research the chosen application's purpose, functionality, and impact.

Analyze how Ethereum's components and smart contracts are utilized in the application.

Discuss the benefits and challenges of using Ethereum in this context.

Write a Detailed Report:

Introduction to the chosen application.

Description of its architecture and workflow.

Analysis of the benefits and challenges.

Conclusion and future prospects.

Verification:

Ensure the report covers all aspects of the case study and provides a thorough analysis of the Ethereum application.

#### Exercise 3:

Implement security measures in smart contracts to prevent common vulnerabilities.

#### Suggested Answer:

Identify Common Vulnerabilities:

Research common smart contract vulnerabilities such as reentrancy attacks, integer overflow/underflow, and access control issues.

Write a Secure Smart Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

import "@openzeppelin/contracts/access/Ownable.sol";

contract SecureContract is ReentrancyGuard, Ownable {

mapping(address => uint256) private balances;

function deposit() public payable {

balances[msg.sender] += msg.value;

}

function withdraw(uint256 amount) public nonReentrant {

require(balances[msg.sender] >= amount, "Insufficient balance");

balances[msg.sender] -= amount;

payable(msg.sender).transfer(amount);

}

function getBalance() public view returns (uint256) {

return balances[msg.sender];

}

}

Verification:

Ensure the contract includes security measures such as reentrancy guards and access control.

Test the contract to verify that the security measures are effective.

#### Exercise 4:

Integrate MetaMask with a DApp to enable users to interact with the Ethereum blockchain.

#### Suggested Answer:

Set Up MetaMask:

Ensure MetaMask is installed and set up with a testnet (e.g., Ropsten).

Connect MetaMask to the DApp:

Modify the DApp interface to detect and connect to MetaMask.

Example code to connect MetaMask:

javascript

Copy code

async function connectMetaMask() {

if (window.ethereum) {

try {

await window.ethereum.request({ method: 'eth\_requestAccounts' });

web3 = new Web3(window.ethereum);

console.log('MetaMask connected');

} catch (error) {

console.error('User denied account access');

}

} else {

console.error('MetaMask not detected');

}

}

connectMetaMask();

Verification:

Ensure the DApp can interact with the Ethereum blockchain through MetaMask, allowing users to perform transactions and interact with smart contracts.

#### Exercise 5:

Develop and deploy an advanced smart contract with complex functionality.

#### Suggested Answer:

Design an Advanced Smart Contract:

Choose a complex use case, such as a decentralized voting system or a multi-signature wallet.

Write the smart contract with detailed functionality.

Example Contract (Multi-Signature Wallet):

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract MultiSigWallet {

address[] public owners;

uint256 public required;

struct Transaction {

address destination;

uint256 value;

bool executed;

}

mapping(uint256 => Transaction) public transactions;

uint256 public transactionCount;

mapping(uint256 => mapping(address => bool)) public confirmations;

constructor(address[] memory \_owners, uint256 \_required) {

require(\_owners.length > 0, "Owners required");

require(\_required > 0 && \_required <= \_owners.length, "Invalid number of required confirmations");

owners = \_owners;

required = \_required;

}

function submitTransaction(address destination, uint256 value) public {

uint256 transactionId = addTransaction(destination, value);

confirmTransaction(transactionId);

}

function addTransaction(address destination, uint256 value) internal returns (uint256) {

transactionCount++;

transactions[transactionCount] = Transaction({

destination: destination,

value: value,

executed: false

});

return transactionCount;

}

function confirmTransaction(uint256 transactionId) public {

require(isOwner(msg.sender), "Not an owner");

require(transactions[transactionId].destination != address(0), "Invalid transaction");

confirmations[transactionId][msg.sender] = true;

executeTransaction(transactionId);

}

function executeTransaction(uint256 transactionId) public {

if (isConfirmed(transactionId)) {

Transaction storage tx = transactions[transactionId];

tx.executed = true;

payable(tx.destination).transfer(tx.value);

}

}

function isConfirmed(uint256 transactionId) public view returns (bool) {

uint256 count = 0;

for (uint256 i = 0; i < owners.length; i++) {

if (confirmations[transactionId][owners[i]]) {

count += 1;

}

}

return count >= required;

}

function isOwner(address account) internal view returns (bool) {

for (uint256 i = 0; i < owners.length; i++) {

if (owners[i] == account) {

return true;

}

}

return false;

}

receive() external payable {}

}

Verification:

Ensure the smart contract has complex functionality and works as intended.

Deploy and test the contract on a testnet.

#### Exercise 6:

Implement a smart contract to track supply chain transactions on Ethereum.

#### Suggested Answer:

Design the Supply Chain Contract:

Write a smart contract to track product movement through the supply chain.

Example Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract SupplyChain {

struct Product {

string name;

address manufacturer;

uint256 timestamp;

string status;

}

mapping(uint256 => Product) public products;

uint256 public productCount;

function createProduct(string memory name) public {

productCount++;

products[productCount] = Product({

name: name,

manufacturer: msg.sender,

timestamp: block.timestamp,

status: "Manufactured"

});

}

function updateStatus(uint256 productId, string memory status) public {

require(products[productId].manufacturer == msg.sender, "Only the manufacturer can update the status");

products[productId].status = status;

}

function getProduct(uint256 productId) public view returns (string memory, address, uint256, string memory) {

Product memory product = products[productId];

return (product.name, product.manufacturer, product.timestamp, product.status);

}

}

Verification:

Deploy the contract and create several products.

Update and track the status of products through the supply chain.

Verify that the contract accurately records and displays product information and status updates.

## Private Study

The time allocation for private study in this topic is expected to be 5 hours.

**Lecturers’ Notes:**

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide. They are expected to also use private study time to review the content of this topic.

You may wish to ask students to complete the private study exercises after the relevant lecture has been delivered and then allow time for review of their answers in the next lecture or to collect in written work for one or more exercises as you feel is appropriate.

#### Exercise 1:

Understand the detailed architecture of Ethereum and its components.

Task:

Research Ethereum Architecture:

Investigate the key components of Ethereum, including nodes, the Ethereum Virtual Machine (EVM), smart contracts, and the consensus mechanism.

Understand the roles of full nodes, light nodes, and miners within the network.

Write a Detailed Report:

Describe the architecture of Ethereum, highlighting the interaction between its components.

Explain how transactions are processed and validated in the Ethereum network.

Discuss the importance of the EVM and how it executes smart contracts.

#### Suggested Answer:

Suggested Report Outline:

Introduction to Ethereum

Overview of Ethereum architecture

Detailed explanation of nodes, EVM, smart contracts, and consensus mechanism

Transaction processing and validation

Conclusion

#### Exercise 2:

Compare and contrast the two leading blockchain platforms, Bitcoin and Ethereum.

#### Suggested Answer:

Research Key Differences:

Study the primary objectives, consensus mechanisms, and functionalities of Bitcoin and Ethereum.

Explore the unique features of each platform, including Bitcoin's focus on digital currency and Ethereum's smart contract capabilities.

Write a Comparative Analysis:

Compare Bitcoin and Ethereum in terms of their architecture, use cases, and performance.

Discuss the pros and cons of each platform.

Highlight real-world applications and how each platform addresses specific needs.

Suggested Report Outline:

Introduction to Bitcoin and Ethereum

Comparison of architecture and consensus mechanisms

Functional differences and unique features

Real-world applications and use cases

Pros and cons of each platform

Conclusion



Topic 5

# Topic 5: Private Blockchain

## Learning Objectives

Private blockchains are restricted networks where access is controlled, typically used by organizations for internal purposes. This chapter explores the characteristics and advantages of private blockchains, focusing on their applications in enterprise environments. It also delves into Hyperledger Fabric, a leading private blockchain framework, and its architecture, components, and use cases in various industries.

On completion of the topic, students will be able to:

* List the key characteristics and necessity of private blockchains. (Knowledge)
* Explain the implementation of smart contracts in a private blockchain environment. (Comprehension)
* Describe state machine replication and consensus algorithms like PAXOS and RAFT. (Application)
* Understand Byzantine Faults and Byzantine Fault Tolerant (BFT) systems. (Analysis)
* Explain the architecture and components of Hyperledger Fabric. (Evaluation)
* Create and manage a Hyperledger Network, with a case study on supply chain management. (Creation)

## Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then investigate the topic further during the laboratory sessions. Private study exercises will be provided to further enhance the students’ understanding of the topic.

## Timings

Lectures: 8 hours Laboratory Sessions: 0 hours Private Study: 8 hours

## Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

* What is private blockchain
* Smart contract in private blockchain
* State machine in smart contract
* Consensus Algorithms for Private Blockchain
* Hyperledger Framework
* Hyperledger tools
* Hyperledger fabric architecture
* Supply Chain Management using Hyperledger

### Guidance on the Use of the Slides

The slides are divided into three lectures, each lasting 1 hour. These may be delivered as eight separate lectures or you may combine them into longer sessions.

The slides introduce cable types and connectors. Where possible, show students real examples of these. If this is not possible, you may wish to find some images to help students envision the different items.

### Lecture 1

|  |  |
| --- | --- |
| Slide 120-123 | What is Private BlockchainA private blockchain is a restricted-access network where participation is controlled by a single organization or a group of organizations. Unlike public blockchains, private blockchains offer enhanced privacy, faster transaction speeds, and more control over data and participants. |

### Lecture 2

|  |  |
| --- | --- |
| Slide 124-127 | Smart Contract in Private BlockchainSmart contracts in private blockchains operate similarly to those in public blockchains but within a controlled environment. They automate and enforce business rules and agreements, ensuring transparency and efficiency in enterprise processes without compromising privacy. |

### Lecture 3

|  |  |
| --- | --- |
| Slide 128-129 | State Machine in Smart ContractA state machine in smart contracts refers to a system that transitions between states based on predefined rules and events. Each state represents a particular condition of the contract, and transitions occur as specific conditions are met, enabling precise control over contract behavior. |

### Lecture 4

|  |  |
| --- | --- |
| Slide 130-145 | Consensus Algorithms for Private BlockchainConsensus algorithms in private blockchains are designed to achieve agreement among trusted nodes with greater efficiency and speed. Common algorithms include Practical Byzantine Fault Tolerance (PBFT), which ensures consistency and fault tolerance, and Raft, which focuses on leader-based consensus for simplicity and performance. |

### Lecture 5

|  |  |
| --- | --- |
| Slide 146-148 | Hyperledger FrameworkThe Hyperledger framework is a collaborative open-source project hosted by the Linux Foundation, aiming to advance cross-industry blockchain technologies. It provides various tools and frameworks for developing enterprise-grade blockchain applications, emphasizing modularity and interoperability. |

### Lecture 6

|  |  |
| --- | --- |
| Slide 149-151 | Hyperledger ToolsHyperledger offers several tools to facilitate blockchain development, including Hyperledger Composer for modeling business networks, Hyperledger Caliper for performance benchmarking, and Hyperledger Explorer for visualizing blockchain data. |

### Lecture 7

|  |  |
| --- | --- |
| Slide 152-157 | Hyperledger Fabric ArchitectureHyperledger Fabric is a modular and extensible blockchain platform designed for enterprise use. Its architecture includes key components like Membership Service Providers (MSPs) for identity management, chaincode for smart contracts, and a consensus mechanism that supports pluggable implementations for flexibility and scalability. |

### Lecture 8

|  |  |
| --- | --- |
| Slide 158-162 | Supply Chain Management using HyperledgerSupply chain management using Hyperledger involves leveraging blockchain technology to enhance transparency, traceability, and efficiency in tracking goods and materials. By using Hyperledger Fabric, companies can create a tamper-proof record of transactions, streamline processes, and reduce fraud and errors in the supply chain. |

## Laboratory Sessions

The laboratory time allocation for this topic is 2 hours.

**Lecturers’ Notes:**

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

## Private Study

The time allocation for private study in this topic is expected to be 5 hours.

**Lecturers’ Notes:**

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide. They are expected to also use private study time to review the content of this topic.

You may wish to ask students to complete the private study exercises after the relevant lecture has been delivered and then allow time for review of their answers in the next lecture or to collect in written work for one or more exercises as you feel is appropriate.

#### Exercise 1:

Gain a comprehensive understanding of the Hyperledger Fabric architecture and its components.

Your report should be approximately 300 words.

#### Suggested Answer:

Task:

Research Hyperledger Fabric Architecture:

Investigate the key components of Hyperledger Fabric, including Membership Service Providers (MSPs), peer nodes, orderer nodes, and chaincode.

Understand the roles of each component in the overall architecture and how they interact to create a secure and efficient blockchain network.

Write a Detailed Report:

Describe the architecture of Hyperledger Fabric, highlighting the function of each component.

Explain how transactions are processed and validated in a Hyperledger Fabric network.

Discuss the consensus mechanisms supported by Hyperledger Fabric and their benefits.

Suggested Report Outline:

Introduction to Hyperledger Fabric

Overview of Hyperledger Fabric architecture

Detailed explanation of MSPs, peer nodes, orderer nodes, and chaincode

Transaction processing and validation in Hyperledger Fabric

Supported consensus mechanisms and their benefits

Conclusion

#### Exercise 2:

Understand the application of Hyperledger Fabric in supply chain management by designing a simple supply chain management system.

Your report should be approximately 200 words.

#### Suggested Answer:

Task:

Research Supply Chain Management with Hyperledger Fabric:

Explore how Hyperledger Fabric can be used to improve supply chain management by providing transparency, traceability, and efficiency.

Investigate existing implementations and case studies to understand best practices and common challenges.

Design a Simple Supply Chain Management System:

Create a conceptual design for a supply chain management system using Hyperledger Fabric.

Define the key components of the system, including the roles of participants (e.g., manufacturers, suppliers, retailers), the types of transactions, and the data to be recorded on the blockchain.

Write a Detailed Report:

Describe the design of the supply chain management system, including the roles of participants and the workflow of transactions.

Explain how Hyperledger Fabric's features (e.g., chaincode, private data collections) are utilized to achieve the system's objectives.

Discuss the benefits and potential challenges of implementing such a system in a real-world supply chain.

Suggested Report Outline:

Introduction to supply chain management with blockchain

Overview of Hyperledger Fabric in supply chain management

Conceptual design of the supply chain management system

Roles of participants and workflow of transactions

Utilization of Hyperledger Fabric features

Benefits and potential challenges

Conclusion



Topic 6

# Topic 6: Tools and Applications of Blockchain

## Learning Objectives

Emerging blockchain platforms like Corda, Ripple, and Quorum offer specialized solutions tailored to different industries and use cases. This chapter explores these platforms, highlighting their unique features, consensus mechanisms, and applications. Additionally, it delves into the rapidly growing field of Decentralized Finance (DeFi), examining how blockchain is transforming financial services through innovative, decentralized solutions.

On completion of the topic, students will be able to:

* Identify the architecture and applications of emerging blockchain platforms like Corda, Ripple, and Quorum. (Knowledge)
* Explain the role of blockchain technology in Decentralized Finance (DeFi). (Comprehension)
* Analyze case studies of blockchain platforms used in DeFi applications. (Application)
* Compare and contrast the advantages and limitations of different emerging blockchain platforms. (Analysis)
* valuate the integration of blockchain technology in various financial and non-financial sectors. (Evaluation)
* Explore future trends and propose potential developments in blockchain technology. (Creation)

## Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then investigate the topic further during the laboratory sessions. Private study exercises will be provided to further enhance the students’ understanding of the topic.

## Timings

Lectures: 3 hours Laboratory Sessions: 4 hours Private Study: 3 hours

## Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

* Hyperledger Fabric
* CORDA and Ripple
* Quorum and DeFi

### Guidance on the Use of the Slides

The slides are divided into two lectures, each lasting 1 hour. These may be delivered as three separate lectures or you may combine them into longer sessions.

### Lecture 1

|  |  |
| --- | --- |
| Slide 164-166 | Hyperledger FabricHyperledger Fabric is a modular and permissioned blockchain framework designed for enterprise use. It supports plug-and-play components, including consensus and membership services, and allows for private and confidential transactions. Hyperledger Fabric is known for its flexibility, scalability, and ability to handle a wide range of business applications. |

### Lecture 2

|  |  |
| --- | --- |
| Slide 167-170 | CORDA and RippleCorda is a blockchain platform specifically designed for businesses to manage complex transactions securely and efficiently. It focuses on privacy and allows for direct transactions between parties. Ripple, on the other hand, is a real-time gross settlement system and remittance network aimed at enabling instant, cross-border payments. Ripple's consensus algorithm is designed to facilitate fast and low-cost international money transfers. |

### Lecture 3

|  |  |
| --- | --- |
| Slide 171-176 | Quorum and DeFiQuorum is an enterprise-focused version of Ethereum that provides high-speed and high-throughput processing of private transactions within a permissioned network. It is optimized for use cases requiring data privacy and high performance. Decentralized Finance (DeFi) refers to financial systems built on public blockchains like Ethereum, allowing for decentralized lending, borrowing, and trading without traditional intermediaries. DeFi leverages smart contracts to provide transparent and trustless financial services accessible to anyone with an internet connection. |

## Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

**Lecturers’ Notes:**

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Students will need to work in small groups to complete the laboratory tasks below. You will need to have facilities available to allow students to create the small network in Exercise 1 below. Following completion of the exercise, you may wish to run a class feedback session of how the network was set up to support students before they undertake the report write up during private study time.

#### Exercise 1:

Implement a simple voting system using smart contracts to understand the application of smart contracts in real-world scenarios.

Suggested Answer:

Writing the Voting Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract Voting {

struct Candidate {

uint id;

string name;

uint voteCount;

}

mapping(uint => Candidate) public candidates;

mapping(address => bool) public voters;

uint public candidatesCount;

constructor() {

addCandidate("Alice");

addCandidate("Bob");

}

function addCandidate(string memory \_name) private {

candidatesCount++;

candidates[candidatesCount] = Candidate(candidatesCount, \_name, 0);

}

function vote(uint \_candidateId) public {

require(!voters[msg.sender], "You have already voted.");

require(\_candidateId > 0 && \_candidateId <= candidatesCount, "Invalid candidate ID.");

voters[msg.sender] = true;

candidates[\_candidateId].voteCount++;

}

}

Deploying the Voting Contract:

Compile the contract in Remix IDE.

Deploy the contract and note the candidate IDs for voting.

Interacting with the Voting Contract:

Use the vote function to cast votes for candidates.

Check the vote counts for each candidate using the candidates mapping.

Verification:

Verify that the voting process works by casting votes and checking the updated vote counts.

Ensure that the contract correctly prevents double voting by the same address.

#### Exercise 2:

Implement a smart contract to track supply chain transactions on Ethereum.

#### Suggested Answer:

Design the Supply Chain Contract:

Write a smart contract to track product movement through the supply chain.

Example Contract:

solidity

Copy code

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract SupplyChain {

struct Product {

string name;

address manufacturer;

uint256 timestamp;

string status;

}

mapping(uint256 => Product) public products;

uint256 public productCount;

function createProduct(string memory name) public {

productCount++;

products[productCount] = Product({

name: name,

manufacturer: msg.sender,

timestamp: block.timestamp,

status: "Manufactured"

});

}

function updateStatus(uint256 productId, string memory status) public {

require(products[productId].manufacturer == msg.sender, "Only the manufacturer can update the status");

products[productId].status = status;

}

function getProduct(uint256 productId) public view returns (string memory, address, uint256, string memory) {

Product memory product = products[productId];

return (product.name, product.manufacturer, product.timestamp, product.status);

}

}

Verification:

Deploy the contract and create several products.

Update and track the status of products through the supply chain.

Verify that the contract accurately records and displays product information and status updates.

## Private Study

The time allocation for private study in this topic is expected to be 3 hours.

**Lecturers’ Notes:**

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide. They are expected to also use private study time to review the content of this topic.

You may wish to ask students to complete the private study exercises after the relevant lecture has been delivered. You may also wish to allow time for review of their answers in the next lecture or collect in written work for one or more exercises as you feel is appropriate.

#### Exercise 1:

Understand the unique features, use cases, and advantages of emerging blockchain platforms such as Hyperledger Fabric, Corda, Ripple, and Quorum, and explore the impact of Decentralized Finance (DeFi) on the financial industry.

#### Suggested Answer:

Research and Compare Platforms:

Investigate the key features, consensus mechanisms, and primary use cases of Hyperledger Fabric, Corda, Ripple, and Quorum.

Understand how each platform is designed to address specific industry needs and what sets them apart from one another.

Analyze DeFi Applications:

Explore the concept of Decentralized Finance (DeFi) and its significance in the blockchain ecosystem.

Identify and analyze popular DeFi applications built on platforms like Ethereum, focusing on their functionalities and benefits.

Write a Detailed Report:

Provide an overview of each blockchain platform, highlighting its unique features and use cases.

Compare and contrast the platforms in terms of their architecture, consensus mechanisms, and industry applications.

Discuss the rise of DeFi, its key components, and how it is revolutionizing traditional financial services.

Conclude with a summary of the findings and potential future trends in blockchain technology and DeFi.

Suggested Report Outline:

Introduction:

Brief overview of emerging blockchain platforms and the importance of DeFi.

Platform Analysis:

Hyperledger Fabric:

Key features and components.

Primary use cases and industry applications.

Corda:

Unique features and privacy focus.

Use cases in financial services and other industries.

Ripple:

Real-time gross settlement and cross-border payments.

Advantages and primary use cases.

Quorum:

Enterprise-focused version of Ethereum.

Use cases requiring data privacy and high performance.

Comparison of Platforms:

Architectural differences.

Consensus mechanisms.

Advantages and limitations of each platform.

Decentralized Finance (DeFi):

Overview of DeFi and its significance.

Key components and popular applications (e.g., Uniswap, Compound, Aave).

Benefits and challenges of DeFi.

Conclusion:

Summary of findings.

Potential future trends in blockchain technology and DeFi.