Foundations of Cryptocurrency and Blockchain

Instructor: Vegard H. Larsen, Department of Data Science and Analytics, BI

Semester: [Fall/Spring 20XX]

Course Overview

This course offers a comprehensive introduction to the foundations of cryptocurrency and blockchain technology. It is designed for master's level business students with some programming experience (especially in Python) but no background in cryptography. The course will have two main parts:

Part A: Cryptography Fundamentals

Over eight lectures, we will balance theoretical concepts with hands-on Python coding exercises. **Students will learn how fundamental cryptographic primitives and data structures enable cryptocurrencies to function, and they will implement a basic cryptocurrency (in the spirit of Bitcoin) from scratch.** The course follows an implementation-focused approach inspired by Andrej Karpathy's open-source project `cryptos`, building a simple Bitcoin-like system in pure Python. We emphasize understanding of fundamental principles (ledgers, hash functions, key cryptography, transactions, consensus, etc.) over optimization or production-ready performance. By the end of the course, students will have constructed a rudimentary blockchain ledger and grasped how decentralization, cryptography, and consensus come together to enable cryptocurrencies.

Part B: The market for cryptocurrencies; the role of blockchain technology in business; forecasting the value of cryptocurrencies

Remaining 4-7 lectures, TBA
Francesco Ravazzolo might contribute to the forecasting part.

Course Structure

Lecture 1: Introduction to Ledgers and Basic Cryptography

- The role of ledgers in financial transactions: From traditional double-entry bookkeeping to distributed ledgers; how consensus on a ledger's state is crucial for trust in financial systems.
- Cryptographic hashes and their properties: Definition of hash functions; properties like one-wayness, collision resistance, and fixed-size output. How hashes ensure data integrity and link blocks in a blockchain.
- **Hands-on:** Implementing a basic hash function in Python: Using Python to compute SHA-256 digests (via a library or simple algorithm) for example data. Students will see how even a tiny change in input yields a vastly different hash, illustrating the avalanche effect.

Resources:

- Narayanan et al. (2016), *Bitcoin and Cryptocurrency Technologies* - Chapter 1 introduces cryptographic hash functions (and basic crypto) in the context of cryptocurrencies.

- UK Government Office for Science (2016), *Distributed Ledger Technology: Beyond Block Chain* - High-level overview of ledgers and blockchain technology, discussing potential applications and challenges (optional reading for broader context).
- Python `hashlib` documentation – Reference for Python's built-in hash functions (used for the hands-on exercise).

Lecture 2: Public and Private Keys, Digital Signatures

- Asymmetric cryptography & key pairs: Introduction to public-key cryptography. How a private key (secret) and public key are generated and related. Role of hard mathematical problems (e.g. elliptic curves) in securing keys.
- Digital signatures and authentication: How to sign a message with a private key and verify with a public key. The importance of digital signatures in proving ownership of funds in a cryptocurrency (owner of a private key can authorize transactions).
- **Hands-on:** Generating keys and signing messages in Python: Students will generate an asymmetric key pair (using an existing Python crypto library or provided code) and produce a digital signature for a sample message. They will then verify the signature, illustrating authenticity and integrity.

Resources:

- Antonopoulos (2017), *Mastering Bitcoin*, 2nd Ed. - Chapter 4 (Keys and Addresses) and Chapter 8 (Digital Signatures) explain how Bitcoin uses elliptic curve cryptography (secp256k1) for keys and ECDSA for signatures, with accessible examples.
- Corbellini, A. (2015), "Elliptic Curve Cryptography: a gentle introduction" - Blog series providing an intuitive walkthrough of elliptic curve math and ECDSA. An excellent supplementary resource for understanding the math behind key generation and signing.
- PyCA Cryptography Library (Python) – A popular Python cryptography library. Documentation includes how to generate RSA/ECC keys and sign/verify messages, useful for the coding exercises.

Lecture 3: Transactions and UTXOs

- Anatomy of a cryptocurrency transaction: Breaking down a Bitcoin transaction structure: inputs (references to past outputs and signatures) and outputs (recipient addresses and amounts). How transactions transfer value by consuming UTXOs and creating new UTXOs.
- UTXO (Unspent Transaction Output) model: Explanation of the UTXO ledger model used by Bitcoin. Each output can be spent at most once, and transactions must reference valid unspent outputs. Comparison with an account model (as used in some other systems) to highlight design differences.
- **Hands-on:** Creating and verifying simple transactions in Python: Students will simulate a basic transaction. Using Python objects or dictionaries, they will create a transaction that consumes a made-up UTXO, verify that the input had sufficient balance, and produce new output(s). They will implement a simple check that prevents double-spending (ensuring the same UTXO is not used twice).

Resources:

- Antonopoulos (2017), *Mastering Bitcoin*, 2nd Ed. - Chapter 6 (Transactions) provides a detailed look at how Bitcoin transactions are structured, including examples of inputs, outputs, and how UTXOs function.
- Bitcoin Developer Guide: Transactions and UTXOs - An official guide explaining the lifecycle of transactions in Bitcoin and how UTXOs are created and spent. Offers insight into transaction validity rules and how wallets select UTXOs.

- Karpathy's `cryptos` (GitHub repository) - An educational from-scratch Bitcoin implementation in Python. The source code (e.g. `transaction.py`) can be referenced to see how transactions and UTXOs are handled in a simplified Bitcoin model.

Lecture 4: Blockchain Data Structures

- Block structure and hash linking: What constitutes a block (block header and list of transactions). Fields like previous block hash, Merkle root, timestamp, nonce, etc. How each block's header includes the hash of the previous block, creating a hash-chain (blockchain) that secures historical data (any tampering breaks the chain).
- Merkle trees and their role in verification: Introduction to Merkle trees (hash trees). How transaction hashes are paired and hashed repeatedly to produce a single Merkle root in the block header. The Merkle root allows efficient membership proofs (Merkle proofs) for transactions, which is essential for light clients (SPV nodes) to verify transactions without downloading full blocks.
- **Hands-on:** Implementing a simple blockchain in Python: Students will write a simple Block class in Python with attributes for index, previous hash, list of transactions, timestamp, etc. They will implement linking blocks by setting each block's previous-hash field to the hash of the prior block. Additionally, they will create a function to compute a Merkle root from a list of transaction hashes (for simplicity, using binary pairing or even a linear hash chain if binary tree is too complex), reinforcing how the data structure works.

Resources:

- Antonopoulos (2017), *Mastering Bitcoin*, 2nd Ed. - Chapter 11 (Bitcoin Blockchain) covers how blocks are structured and chained, and explains Merkle trees with diagrams and examples. It provides a clear picture of block headers and the function of the Merkle root.
- Investopedia: Merkle Root & Trees - A non-technical explanation of Merkle trees in the context of blockchain. Useful for grasping the concept of how transaction hashes combine into a single root and why that matters for security and efficiency.

Lecture 5: Consensus Mechanisms

- Why consensus is needed: Discussion of the double-spending problem in a decentralized network. The need for a mechanism to agree on a single transaction history (canonical chain) without a central authority. Overview of how new blocks are propagated and conflicts (forks) are resolved by consensus rules.
- Proof of Work (PoW) vs. Proof of Stake (PoS): In-depth look at PoW as used in Bitcoin: miners compete to solve a hash puzzle (find a nonce so that block hash ¡ target). How difficulty adjusts, and why PoW makes attacks expensive. Introduction to PoS as an alternative: validators stake cryptocurrency and are chosen to create blocks (basic idea of staking and security assumptions, e.g. the concept of "slashing" and weak subjectivity). Compare the energy usage, security assumptions, and attack vectors of PoW and PoS.
- **Hands-on:** Implementing a basic PoW in Python: Students will be given or write a simple mining loop that tries different nonce values to find a hash with a certain number of leading zeros (simulating a difficulty target). This exercise demonstrates the computational effort required by PoW. They will adjust the difficulty (number of zeros) to see the effect on mining time. (Note: This is purely for demonstration—actual Bitcoin uses a much larger space and double SHA-256.)

Resources:

- Narayanan et al. (2016), *Bitcoin and Cryptocurrency Technologies* - Chapters on mining and consensus (e.g., Chapter 5) explain how Bitcoin's Proof of Work consensus achieves agreement and security, and discuss attacks like 51% attacks.

- Nakamoto, S. (2008), Bitcoin: A Peer-to-Peer Electronic Cash System - The original Bitcoin whitepaper (sections 3 and 4) introduces the concept of Proof of Work consensus to secure the blockchain. A seminal reference for understanding why PoW solves the double-spending problem (advanced reading).
- Buterin, V. (2017), "Proof of Stake FAQ" - An approachable Q&A-style explanation of Proof of Stake by Ethereum's creator. Discusses the motivation for PoS, how it differs from PoW, and addresses common questions and misconceptions about PoS (recommended for a conceptual grasp of PoS).

Lecture 6: Networking and Peer-to-Peer Communication

- Decentralization and P2P networks: How blockchain networks are structured as peer-to-peer networks rather than client-server. Nodes independently hold the ledger and connect to multiple peers. Advantages of P2P (no single point of failure, censorship resistance) and challenges (consistency, propagation delays).
- Gossip protocol and node discovery: Overview of how messages (transactions, blocks) propagate through the network via gossip (each node relays info to its peers). Introduction to how nodes find each other: peer discovery mechanisms, addressing (IP/port), and the role of protocols like DNS seeds or bootstrap nodes in Bitcoin.
- **Hands-on:** Setting up a simple P2P network in Python: Students will simulate a tiny peer-to-peer network. For instance, run multiple instances of a simple Python script (or threads) that communicate. They will use Python's socket library (or a higher-level networking library) to have nodes send messages to each other (e.g., a "transaction" string). The exercise could involve each node relaying a received message to its other peers, illustrating gossip. (The demo can be local on one machine or within the classroom network.)

Resources:

- Antonopoulos (2017), *Mastering Bitcoin*, 2nd Ed. - Chapter 10 (Network and Peer-to-Peer) describes how the Bitcoin P2P network is structured, including message relay, inventory propagation, and the peer discovery process. It provides context for the design of networking in blockchains.
- Python Socket Programming HOWTO - A tutorial on using Python's socket module for network communication. Useful as a reference for the hands-on exercise, helping students understand how to send/receive data between nodes.

Lecture 7: Smart Contracts and Layer 2 Scaling

- Basics of smart contracts and Ethereum's Virtual Machine: Transition from Bitcoin's limited scripting to Ethereum's general-purpose blockchain. What smart contracts are - self-executing code stored on the blockchain. Introduction to the Ethereum Virtual Machine (EVM) that runs contract code on every node. High-level overview of how contracts are deployed and executed, and the concept of gas for computational cost.
- Overview of Layer 2 solutions: Two major approaches to scaling beyond Layer 1:
  - Lightning Network (Bitcoin): Payment channels and state channels. How the Lightning Network allows many small transactions off-chain, settling on-chain only if necessary. Basic channel mechanism (opening, updating balances off-chain, closing with final settlement or breach remedy).
  - Rollups (Ethereum): Idea of bundling or "rolling up" many transactions off-chain and posting a compressed proof or summary on-chain. Difference between optimistic rollups (assuming validity, with fraud proofs) and zk-rollups (zero-knowledge proofs for validity). Benefits of moving computation off-chain while relying on Layer 1 for security.

- **Hands-on:** Writing a simple smart contract in Python (simulated): Instead of Solidity, students will use a Python environment to simulate a smart contract. For example, implement a simple escrow logic or token transfer as a Python function/class to illustrate state and functions (or use a framework like `brownie` or `web3.py` to interact with an actual simple Solidity contract provided by the instructor). The goal is to expose students to the concept of writing contract logic and (if possible) executing it in a test blockchain environment. (Note: This exercise will be kept basic due to limited time and the audience's background. It might involve using a pre-deployed contract or a local Ethereum simulator.)

Resources:

- Antonopoulos & Wood (2018), *Mastering Ethereum* - An open-source book covering Ethereum's architecture and smart contracts. Chapters 7 and 8 introduce smart contract programming (in Solidity and Vyper), and Chapter 13 discusses the Ethereum Virtual Machine. Provides background for understanding how smart contracts work under the hood.
- Web3.py Documentation - Web3.py is a Python library for interacting with Ethereum. Its docs include tutorials on deploying contracts and calling contract functions from Python. This can be used for the hands-on portion if we interact with a real or local Ethereum node.
- Investopedia: Lightning Network - Introductory article explaining the Bitcoin Lightning Network in simple terms. Helps students grasp the purpose of payment channels and how Lightning achieves fast, low-cost transactions by handling them off-chain.
- Buterin, V. (2021), "An Incomplete Guide to Rollups" - Blog post by Vitalik Buterin giving an overview of Layer 2 rollups on Ethereum. It compares different Layer 2 approaches (state channels, Plasma, optimistic rollups, zk-rollups) and is useful for students to read about current scaling techniques beyond the basic concepts (optional, for deeper insight).

Lecture 8: Final Project and Ethical Considerations

- Project guidelines and expectations: An outline of the semester-long group project. Students (in teams of 3-5) will propose and develop a project that integrates concepts from all lectures. Example project ideas: implement a simplified cryptocurrency with novel feature X, build a small decentralized application on a test blockchain, analyze and report on performance of PoW vs PoS under certain assumptions, etc. The project should involve both coding and conceptual analysis. Key milestones (proposal, midpoint progress, final presentation) and due dates will be clarified.
- Ethical concerns in blockchain technologies: Group discussion on broader implications:
  - Privacy: The balance between transparency and anonymity in blockchain. Bitcoin's pseudonymity and the risk of deanonymization; techniques like coin mixing or privacy-centric coins (e.g. Monero) and their controversies.
  - Regulation: How cryptocurrencies challenge existing financial regulations (KYC/AML). Discussion of global regulatory perspectives, such as securities law for ICOs, central bank stances on crypto, and the emergence of CBDCs (central bank digital currencies) in response.
  - Sustainability: The environmental impact of Proof of Work mining (energy consumption, carbon footprint). Exploration of whether Proof of Stake and other alternatives effectively mitigate these issues. Also, consideration of e-waste from mining hardware and efforts to move towards greener blockchain technologies.
- **Hands-on:** Presenting and refining project ideas: Each team will give a brief presentation of their project proposal (problem they want to tackle, approach, and division of tasks). Class and instructor will provide feedback and suggestions, with a focus on ensuring the project is scoped appropriately and covers relevant course concepts. Teams will refine their plans accordingly. (This session is interactive and meant to solidify project direction.)

Resources:

- Narayanan et al. (2016), *Bitcoin and Cryptocurrency Technologies* - Chapter 6 (Bitcoin and Anonymity) and Chapter 7 (Community, Politics, and Regulation) discuss privacy issues in cryptocurrencies and the regulatory/political landscape. These provide a scholarly perspective on the ethical topics discussed.
- de Vries, A. (2018), "Bitcoin's Growing Energy Problem" - An academic article in Joule examining Bitcoin's energy consumption in detail. It quantifies the sustainability concerns and is a starting point for discussing environmental impact (students can read the introduction/conclusion for key points).
- (Various news articles and reports provided on the course site) - Throughout the semester, links to current articles on regulatory developments, environmental news, or major security incidents in crypto will be shared to enrich discussions in this lecture. (For example, recent SEC statements on crypto, or improvements in miner renewable energy usage, etc.)

Assessment and Final Project

Semester-Long Group Project: The primary assessment is a comprehensive group project (teams of 3-5 students) that spans the duration of the course. The project should integrate key concepts from all lectures, demonstrating both theoretical understanding and practical implementation skills. Example project themes include: developing a mini cryptocurrency with a unique feature, creating a prototype decentralized application or smart contract system, analyzing and visualizing blockchain data, or experimenting with a consensus algorithm variation. Students are encouraged to be creative while staying within scope.
Deliverables:

- **Project Proposal:** Due by the mid-semester point (around Lecture 5 or 6). A brief document (1-2 pages) outlining the project idea, objectives, methodology, and a work plan. Teams will present these proposals during Lecture 8 to get feedback.
- **Final Code & Documentation:** A repository (e.g., on GitHub) containing well-documented Python code developed for the project. This should include a README or report (5-10 pages) explaining the design, how to run the code, and how course concepts are implemented. Any tests or demonstration scripts should be provided to show the project in action.
- **Presentation/Demo:** In the final week of the semester (or exam week), each group will deliver a 15-20 minute presentation. This will include a live demo of their project (if applicable) and a discussion of challenges faced, lessons learned, and how course principles are illustrated.

Evaluation Criteria:

- Conceptual Depth: How well the project incorporates and demonstrates understanding of the course topics (cryptography, transactions, consensus, etc.). Projects should clearly connect to at least several core concepts from the lectures.
- Implementation: The functionality and correctness of the code. While not focusing on optimization, the code should be readable, reasonably efficient for the task, and properly documented. Innovative or elegant technical solutions will be viewed favorably.
- Team Collaboration: Evidence of effective teamwork, such as clear division of tasks, and a coherent integration of components. Peer evaluations may be used to ensure equitable contributions.
- Presentation and Communication: Clarity and organization of the final presentation and written report. The ability to explain the project's purpose, design choices, and results to both a technical and non-technical audience (remembering that as business students, communicating complex tech in simple terms is a valuable skill).
- Originality and Difficulty: Ambitious or creative projects that go beyond class examples (within reason) can earn extra credit. However, even a simple project executed flawlessly and explained well can score high. The scope should be balanced with the time frame.

There is no traditional exam in this course. Instead, the focus is on continuous learning through the project and participation in lectures. Participation is expected: students should engage in discussions, ask questions, and contribute during hands-on coding sessions and group activities. By the end of the course, students will have both a theoretical foundation in how cryptocurrencies and blockchains work and practical experience from having built elements of these systems themselves.