Poznan University of Technology

Blockchain Technology And Quantum Computation
Blockchain Mechanisms

Jakub Piotr Hamerliński

Blockchain Technology And Quantum Computation
Blockchain Mechanisms

01. Cryptographic mechanisms used in blockchain technology: Proof-of-Work, Proof-of-Stake, Byzantine Fault Tolerance.
02. Smart contracts and their role in blockchain-based applications.
03. Designing a blockchain structure for a specific application.
04. Hash functions and digital signatures in blockchains
05. Secure multi-party computation
06. Task

Proof-of-Work
Proof-of-Stake
Byzantine Fault Tolerance

Blockchain Mechanisms

Proof-of-Work

Proof-of-Work (PoW): Proof-of-Work is a consensus algorithm used in many cryptocurrencies, most notably Bitcoin. In a PoW system, participants (known as miners) compete to solve a complex mathematical problem that requires significant computational resources. The first miner to solve the problem gets the right to add a new block of transactions to the blockchain and is rewarded with a certain amount of cryptocurrency. The problem is difficult to solve but easy for others to verify once a solution is found. PoW systems are designed to be resistant to attacks because altering the blockchain would require redoing all the work of block creation, which is computationally expensive and time-consuming.

Blockchain Mechanisms

Proof-of-Work - example

01. A new set of transactions is bundled into a block.
02. Miners take the information in this block and perform computations on it to find a value (nonce) that, when hashed along with the block data, produces a hash with a specific number of leading zeros. This specific number is determined by the network's difficulty level.
03. This is a trial-and-error process, and the complexity of the problem ensures that it takes a considerable amount of computational power and time to find the correct nonce.
04. The first miner to find a nonce that satisfies the condition broadcasts the block to the network.
05. Other miners verify the solution. If it's correct, they add the block to their version of the blockchain and start working on the next block.

Blockchain Mechanisms

Proof-of-Stake

Proof-of-Stake (PoS): Proof-of-Stake is another type of consensus algorithm used in some cryptocurrencies, such as Ethereum. In a PoS system, the creator of a new block is chosen in a deterministic way, depending on the participant's stake, or ownership of the cryptocurrency. Participants are randomly selected to validate blocks, and the probability of being chosen is proportional to the amount of currency the participant holds. PoS systems are generally considered to be more energy-efficient than PoW systems, as they don't require miners to perform complex calculations.

Blockchain Mechanisms

Proof-of-Stake - example

01. In a PoS system, validators are chosen to create a new block, typically based on the amount of cryptocurrency they hold and are willing to "stake" as collateral.
02. When a new block is needed, the protocol will select one of the staking nodes to validate the next block. The likelihood of being chosen is generally proportional to the number of coins a node has staked.
03. The chosen validator checks and confirms the transactions within the block, signs the block, and adds it to the blockchain.
04. Validators are rewarded with transaction fees or in some cases, a new coin.
05. If a validator tries to attack the network or approves fraudulent transactions, their stake is forfeited. This provides a strong incentive for validators to act honestly.

Blockchain Mechanisms

Byzantine Fault Tolerance

Byzantine Fault Tolerance (BFT): Byzantine Fault Tolerance is a property of a system that allows it to reach consensus (agreement across all nodes) even when some nodes fail to respond or respond with incorrect information. This is often referred to as the Byzantine Generals Problem, a situation where nodes must agree on a concerted strategy to avoid catastrophic system failure, but some nodes are unreliable. There are different variations of BFT in use in various blockchain systems, including Practical Byzantine Fault Tolerance (PBFT), used by cryptocurrencies like Stellar, and Delegated Byzantine Fault Tolerance (DBFT), used by NEO. In these systems, a group of trusted nodes validates transactions and blocks. These validators are chosen by different methods, depending on the specific BFT system.

Blockchain Mechanisms

Byzantine Fault Tolerance - example

01. BFT consensus mechanisms involve a group of nodes, known as validators, which take turns proposing and voting on the next block.
02. A node proposes a new block. Other nodes in the network then vote on whether to accept the block or not.
03. If a supermajority (often 2/3 or more) of nodes agree with the proposed block, it is added to the blockchain.
04. The system continues to function correctly and reach consensus as long as the number of malicious nodes doesn’t reach the threshold that could disrupt the supermajority. The threshold is determined by the specific BFT variant (e.g., Practical BFT, Delegated BFT).

Smart contracts

Blockchain Mechanisms

Smart contracts

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They exist across a distributed, decentralized blockchain network, and they permit trusted transactions and agreements to be carried out among anonymous parties without the need for a central authority, legal system, or external enforcement mechanism.

Blockchain Mechanisms

Smart contracts - importance

Here is a brief outline of the role of smart contracts in blockchain-based applications:

01. **Automation**: Smart contracts automate tasks and functions. When certain conditions in the contract are met, the smart contract automatically executes the corresponding contractual clause.
02. **Trust**: The terms and conditions of smart contracts are transparent and visible to all relevant parties. There is no need to trust a single party since the contract self-executes and the output is validated by everyone on the network.

Blockchain Mechanisms

Smart contracts - importance

03. **Security**: Smart contracts are stored on a blockchain, which is decentralized and encrypted. This makes them resistant to hacking and fraud. If any attempt is made to alter the contract, it would be rejected by the network.
04. **Speed and Efficiency**: By automating processes and reducing the need for intermediaries, smart contracts can speed up business operations and make them more efficient.
05. **Savings**: Smart contracts eliminate the need for intermediaries, which can result in significant cost savings.

Blockchain Mechanisms

Smart contracts - usage

Smart contracts are used in many blockchain-based applications, including but not limited to:

01. **Financial services**: Smart contracts can automate various financial processes, such as issuing and trading securities, settling trades, and executing contractual agreements.
02. **Supply chain management**: Smart contracts can automatically track and verify the movement of goods across the supply chain, ensuring transparency and reducing fraud.
03. **Decentralized Autonomous Organizations (DAOs)**: These are organizations that are run by rules encoded in smart contracts. They allow for a governance system that is not controlled by a central authority.

Blockchain Mechanisms

Smart contracts - usage

04. **Real Estate**: Smart contracts can automate the process of buying and selling property, including verifying the identity of the parties, confirming payment, and transferring ownership.
05. **Decentralized Finance (DeFi)**: This is a financial system built on blockchain technology that uses smart contracts to create a transparent and open financial network. DeFi applications use smart contracts to create protocols that replicate existing financial services in a more open, interoperable, and transparent way.

It's important to note that while smart contracts have significant potential, they also have limitations and challenges, such as the difficulty of managing contractual disputes and the risk of bugs in the contract code.

Designing a blockchain structure

Blockchain Mechanisms

Designing a blockchain structure

Designing a blockchain structure for a specific application requires careful consideration of the use case, the required performance, and the desired level of decentralization and security. Here's a basic outline of the steps you might take to design a blockchain structure:

01. **Define the Use Case**: What problem is the blockchain intended to solve? Who will use it, and why? The answers to these questions will guide your design process. For example, a blockchain for supply chain management might need to handle different types of data and have different performance requirements than one designed for a voting system.
02. **Choose a Consensus Mechanism**: The consensus mechanism determines how transactions are validated and how new blocks are added to the blockchain. Different consensus mechanisms have different strengths and weaknesses. For example, Proof of Work (PoW) provides a high level of security but requires a lot of computational power. On the other hand, Proof of Stake (PoS) and Delegated Proof of Stake (DPoS) are more energy-efficient but may be less decentralized.

Blockchain Mechanisms

Designing a blockchain structure

03. **Design the Blockchain Structure**: Will your blockchain be public (anyone can participate) or private (only authorized participants)? Will it be permissionless (anyone can validate transactions) or permissioned (only certain nodes can validate transactions)? Will it be a single chain of blocks, or will it include side chains or sharding to improve scalability?
04. **Define the Transaction Model**: What kind of transactions will the blockchain handle? Will it use a UTXO (Unspent Transaction Output) model like Bitcoin, or an account/balance model like Ethereum? Will it support smart contracts, and if so, what programming language will they use?
05. **Design the Cryptography**: You'll need to choose cryptographic algorithms for hashing (creating a unique identifier for each block), signing transactions (ensuring they can't be tampered with), and possibly for other purposes like zero-knowledge proofs (allowing users to prove they know a value without revealing the value itself).

Blockchain Mechanisms

Designing a blockchain structure

06. **Plan for Scalability and Interoperability**: How will your blockchain handle increasing numbers of transactions? Can it interact with other blockchains or with traditional systems?
07. **Consider Regulatory and Compliance Issues**: Depending on your use case, your blockchain might need to comply with laws and regulations related to data privacy, financial transactions, etc.

Remember, designing a blockchain is a complex process that requires expertise in cryptography, software development, and system design. And because blockchains are decentralized and immutable, mistakes can be costly to fix, so it's crucial to get the design right.

Hash functions and digital signatures in blockchains

Blockchain Mechanisms

Hash Functions

01. A hash function is a function that takes an input (or 'message') and returns a fixed-size string of bytes, typically a digest that is unique to each unique input. It is designed to be a one-way function, in that it’s infeasible to regenerate the original data given only the hash output.
02. In the context of blockchain, each block contains a hash of the previous block, thus creating a chain of blocks. This means that if someone attempts to alter a block, this would change the hash of the block, which in turn would break the chain.
03. Hash functions are also used in creating a unique identifier for the transactions in the block. All the transactions are hashed into a Merkle tree (or hash tree), and the Merkle root is stored in the block header. This allows for efficient and secure verification of the contents of large data structures.

Blockchain Mechanisms

Digital Signatures

01. A digital signature is a cryptographic tool used to verify the authenticity and integrity of a message, software, or digital document. It’s the digital counterpart of a handwritten signature or stamped seal, but it offers far more inherent security.
02. In blockchains, digital signatures are used to verify the authenticity of the sender of a transaction. When a sender initiates a transaction, they create a digital signature by combining their private key with the transaction data and applying a cryptographic algorithm.

Blockchain Mechanisms

Digital Signatures

03. The resulting digital signature is attached to the transaction data and sent across the network. The receivers use the sender's public key to verify the authenticity of the transaction. If it's successfully verified, the transaction is considered authentic and can be added to a block. If the transaction data has been tampered with, the digital signature verification will fail.
04. The use of digital signatures ensures non-repudiation, meaning that a party cannot deny the authenticity of their signature on a document or the sending of a message that they originated.

Together, hash functions and digital signatures provide the fundamental security backbone for blockchain technology, ensuring data integrity, authentication, and non-repudiation.

Secure Multi-Party Computation

Blockchain Mechanisms

Secure Multi-Party Computation

Secure Multi-Party Computation (SMPC) is a subfield of cryptography that enables multiple parties to compute a function over their inputs while keeping those inputs private. In other words, SMPC protocols allow a group of users to collaboratively compute a function over their inputs while keeping those inputs confidential.

Blockchain Mechanisms

Secure Multi-Party Computation

Here’s a simple example to illustrate the concept: suppose you and your friends want to calculate your average salary, but none of you want to reveal your actual salary to the others. Using an SMPC protocol, you could all submit your salaries to the computation in an encrypted form. The computation would be carried out on the encrypted data, and only the final result (the average salary) would be decrypted and revealed.

Blockchain Mechanisms

Secure Multi-Party Computation

SMPC has many potential applications, such as:

01. Privacy-preserving data mining: Companies could use SMPC to collaboratively analyze their combined data without revealing sensitive information to each other.
02. Secure voting systems: SMPC could allow votes to be tallied without revealing who voted for whom.
03. Privacy-preserving machine learning: SMPC could allow a machine learning model to be trained on data from multiple sources without revealing the individual data points.
04. SMPC is an active area of research and many practical challenges still need to be addressed, such as efficiency (SMPC protocols are typically much slower than regular computations), robustness (the protocol needs to handle parties that drop out or try to sabotage the computation), and security (ensuring that no information is leaked even if some of the parties are malicious).

Task

Blockchain Mechanisms

Task

Task: Role-Play Blockchain Consensus Mechanisms and a Smart Contract

Objective: Understand and apply basic concepts of consensus mechanisms and smart contracts in blockchain technology.

Instructions:

01. As a group of four, choose one of the three consensus mechanisms: Proof-of-Work, Proof-of-Stake, or Byzantine Fault Tolerance.

02. Role-play a scenario where you are nodes in a blockchain network using your chosen consensus mechanism. Take turns proposing blocks (which can just be simple numbers or strings for this exercise), and use your consensus mechanism to decide whether to add each block to your chain:

For the **Proof-of-Work**, you could flip a coin to simulate the computational effort (heads, you found the nonce; tails, you didn't). The first to flip heads proposes the next block.

Blockchain Mechanisms

Task

For **Proof-of-Stake**, you could draw lots to determine which one has the right to propose the next block. Write your names on slips of paper and draw from a hat.

For **Byzantine Fault Tolerance**, each of you should propose a block and then vote on which one to add to the chain. The block with the most votes is added.

03. Next, design a simple smart contract related to the tasks in a project or a study group.

Blockchain Mechanisms

Task

Here some examples of smart contracts that mimic real-life scenarios and can be executed within a classroom setting in about 30 minutes:

01. Invoice Contract: A smart contract could be set up such that once a certain block is added to the chain, one group member must issue an "invoice" to another member. The "invoice" could be a simple written request for a favor, a study resource, or something similar. Once the "invoice" is "paid" (the favor is done or the resource is provided), the contract is fulfilled.
02. Bounty Contract: Set up a smart contract that posts a "bounty" for a quick problem-solving task or a trivia question. When a certain block is added, the bounty is activated. The first group member to solve the problem or answer the question collects the bounty. The bounty could be something simple like getting first turn in the next activity, or choosing the topic for the next group discussion.

Blockchain Mechanisms

Task

03. Resource Allocation Contract: This smart contract could specify that when a certain block is added, a group member has to allocate a "resource" among the group. The "resource" could be study tasks for an upcoming exam or roles for a group project. The allocation should be agreed upon by all group members for the contract to be fulfilled.
04. Verification Contract: This smart contract could stipulate that once a certain block is added, one group member must "verify" a statement made by another member. The statement could be a fact about blockchain technology or something learned in class. The "verification" could involve quickly looking up the information online or in a textbook. Once the statement is verified, the contract is fulfilled.

Remember, the goal here is to mimic the functionality of smart contracts in a real-world context, while keeping it manageable within a classroom setting. The focus should be on understanding how smart contracts work, rather than the specifics of the tasks being performed.

05. Execute your smart contract, making sure to follow its rules exactly. If the terms of the contract are met, the action must be performed.

Blockchain Mechanisms

Task

Options for completing this task:

01. **On paper**: Write down your blocks and smart contracts, and keep track of your blockchain on paper.

02. **Using a whiteboard**: Allows for a larger, more visible display.

03. **In a computer document**: You could use a word processor, spreadsheet, or presentation software to create a visual representation of your blockchain and document your smart contract.

04. **Programming simulation**: More advanced groups could write a simple program to simulate their consensus mechanism and smart contract. This could be done in any programming language you're comfortable with.

Remember, this exercise is a simplification and doesn't cover all aspects of these topics. But it should give you a basic understanding of how consensus mechanisms and smart contracts work.

Sources

Proof-of-Work, Proof-of-Stake, and Byzantine Fault Tolerance:

01. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. This is the original Bitcoin whitepaper, where the concept of proof-of-work was first introduced.
02. King, S., & Nadal, S. (2012). PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. This paper introduces the concept of proof-of-stake.
03. Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3), 382-401. This is the original paper that describes the Byzantine Generals Problem, which Byzantine Fault Tolerance aims to solve.

Smart Contracts:

01. Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks. First Monday, 2(9). This paper by Nick Szabo, who coined the term "smart contracts," discusses the concept in detail.

Sources

Hash Functions and Digital Signatures:

01. Katz, J., & Lindell, Y. (2014). Introduction to Modern Cryptography (2nd ed.). CRC Press. This book provides a comprehensive introduction to the modern study of computer security, including hash functions and digital signatures.

Secure Multi-party Computation:

01. Goldreich, O. (2004). Foundations of Cryptography: Volume 2, Basic Applications. Cambridge University Press. This book has a section on Secure Multi-party Computation that provides a theoretical background on the topic.

Thank you

Feel free to reach me via LinkedIn

Fin