# Lesson 2b: DLT Architecture

## Module A: Blockchain Foundations

BSc Blockchain & Cryptocurrency

University Course

2025

# Building the Blockchain

*[COMIC: Block stacking like Tetris]*

*"Each block must fit perfectly with the one below—*
*one wrong piece and the whole structure is compromised."*

**Today's Challenge:** How do we structure data so thousands of computers can verify it independently?

# Learning Objectives

By the end of this lesson, you will be able to:

1. Describe the structure and components of a blockchain block
2. Explain Merkle trees and their role in transaction verification
3. Differentiate between node types: full nodes, light nodes, miners, validators
4. Contrast permissioned and permissionless blockchain architectures

**Prerequisites:** L02a - DLT Fundamentals

*This is Part 2 of 2 on DLT concepts. Part 1 (L02a) covered DLT definition and Byzantine consensus.*

**Building on L02a:** DLT Fundamentals

# The Problem: How do we structure data for trustless verification (no third-party trust required)?

*Part 2/2: DLT Architecture (Integration)*

**The Challenge**

How do we structure blocks and verify transactions efficiently across thousands of nodes without a central database?

**Why It Matters**

- Block structure determines security properties and performance limits
- Bitcoin's block size debate (2015-2017) led to SegWit upgrade and Lightning Network

**What We Need**

- Trustless verification mechanism
- Efficient data structures for tamper-evident storage at scale

**The Cryptoeconomics Question**

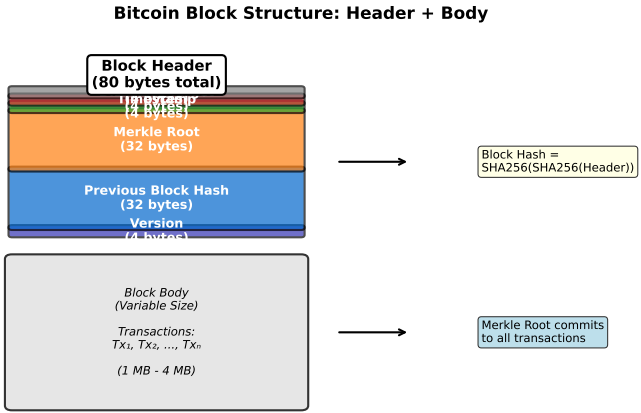
*Coordinating without trusted intermediaries*

*Today's lesson: How DLT architecture addresses this challenge through Merkle trees and node design*

---

**Compare the approaches shown above**

# What is Inside a Block?




**Block Size Limits:** Bitcoin: 1 MB (base block) to 4 MB (with SegWit) — Ethereum: Dynamic gas limit (≈ 30M gas, ≈ 15 MB)

**Block structure provides tamper-evident data containers that enable trustless verification without central coordination**

# What Are the Components of a Block Header?

**Bitcoin Block Header Fields (80 bytes):**

1. **Version** (4 bytes): Block version number (for protocol upgrades)
2. **Previous Block Hash** (32 bytes): Hash of the previous block header
3. **Merkle Root** (32 bytes): Hash of all transactions in this block
4. **Timestamp** (4 bytes): Unix epoch time (seconds since Jan 1, 1970)
5. **Difficulty Target** (4 bytes): Required difficulty for PoW
6. **Nonce** (4 bytes): Counter used for PoW mining

**Key Insight:**

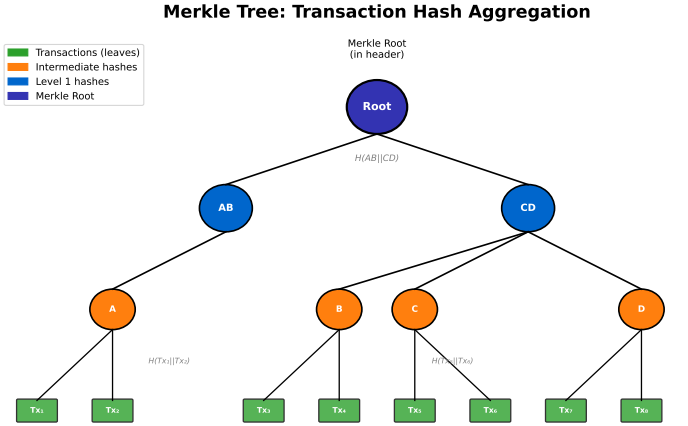
$$\text{Block Hash} = \text{SHA256}(\text{SHA256}(\text{Header}))$$

The hash must be below the difficulty target for the block to be valid.

---

**Key point: Bitcoin Block Header Fields (80 bytes)**

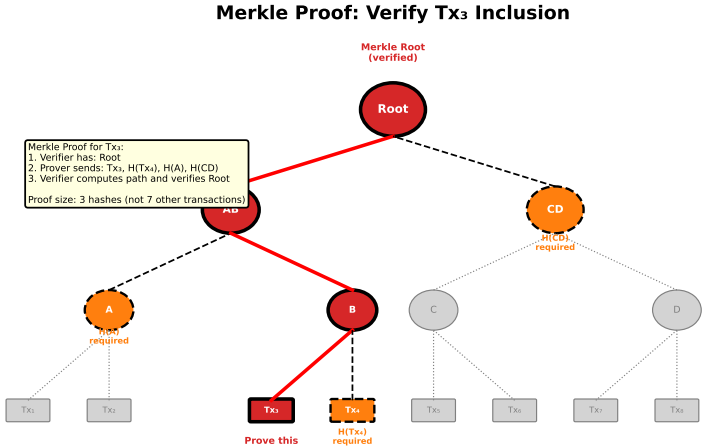
# How Do Merkle Trees Enable Efficient Verification?




**Benefits:** Compact Proof ($O(\log n)$ hashes), Efficient Storage (light clients store only headers), Integrity (any change alters Merkle Root)

**Merkle trees enable light clients to verify transaction inclusion without downloading all transactions**

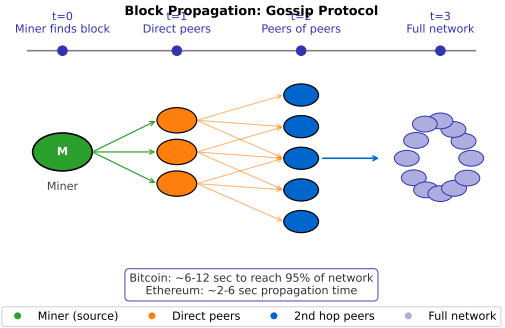
# How Do Merkle Proofs Work?




*Only 3 hashes needed instead of downloading all 8 transactions*

**Merkle proofs enable SPV (Simplified Payment Verification) for light clients**

# How Do Blocks Propagate Through the Network?




**Gossip protocol steps:**

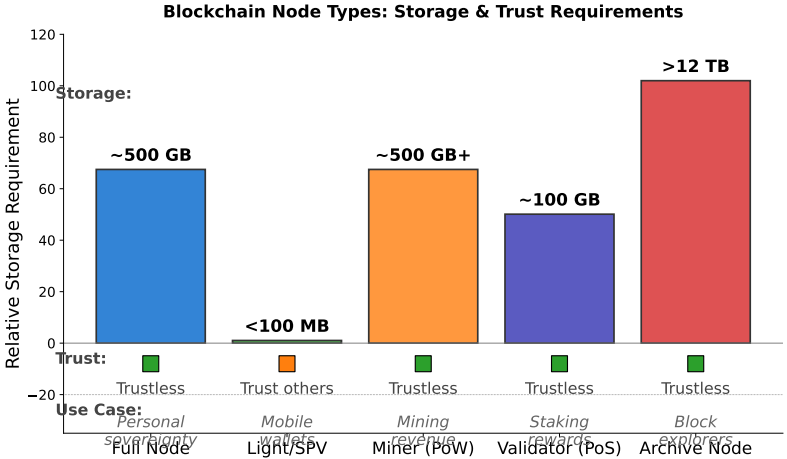
1. Miner finds valid block (solves PoW puzzle)
2. Broadcasts to connected peers ($\approx$ 8-125 peers)
3. Peers validate (hash, transactions, double-spends), then re-broadcast
4. Process repeats until entire network has the block

**Propagation time:**

- Bitcoin: $\approx$ 6-12s to reach 95% of network
- Ethereum (pre-merge): $\approx$ 2-6s
- Faster propagation = fewer orphan blocks

**Gossip protocol enables distributed propagation without central coordination; latency drives orphan/fork rate**

# What Types of Nodes Exist in Blockchains?




**Trade-off:** Higher storage requirements enable trustless verification; lower requirements require trusting others.

# What Do Full Nodes Do?

## Full Node Definition

A **full node** downloads and validates the entire blockchain history, enforcing all consensus rules without relying on third parties.

**Responsibilities:**

- Store complete blockchain (Bitcoin: $\approx$ 600 GB, Ethereum: $\approx$ 2 TB)
- Validate all blocks and transactions independently
- Relay valid transactions and blocks to peers
- Serve blockchain data to light clients

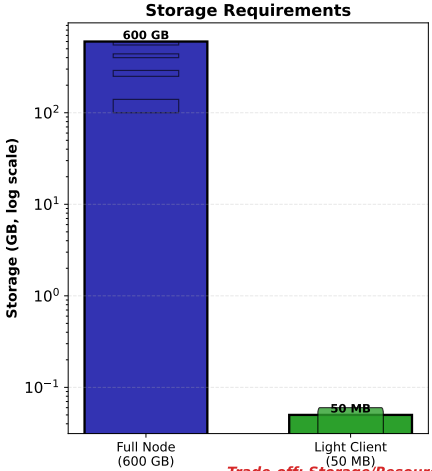
**Benefits:**

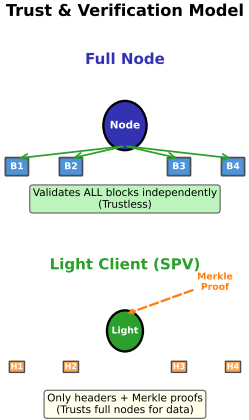
- Maximum security (trust no one)
- Support network decentralization
- Can verify historical transactions

**Requirements:** $\approx$ 500 GB storage, $\approx$ 5 GB/day bandwidth, moderate CPU

**Key point: full node**

# Full Nodes vs. Light Clients: The Trust-Storage Trade-off




***Trade-off: Storage/Resources ↔ Trust/Security***

12,000x storage difference creates accessibility barrier determining network decentralization

**Light clients enable broader participation but introduce trust assumptions**

# Problem Check-In: DLT Architecture

**Recall Our Problem**

*How do we reach agreement in unreliable networks?*

**What We've Learned So Far**

- Block headers enable efficient verification (80 bytes vs full block)
- Node diversity distributes trust across the network
- Both enable scalable, trustless architecture

**Still to Address**

- How do permissioned vs permissionless architectures differ?
- What are the storage/bandwidth trade-offs for different node types?

**Think About**

- Based on what you've seen, how would *you* solve this problem?
- What trade-offs do you expect?

**Pause and reflect: How does what we've learned so far address "How do we reach agreement in unreliable..."?**

# How Do Miner Nodes Work in Proof-of-Work?

**Specialized Nodes in Proof-of-Work Blockchains**

**Functions:**

1. Collect unconfirmed transactions from mempool
2. Construct candidate block (select transactions based on fees)
3. Attempt to find valid nonce (compute millions of hashes/second)
4. Broadcast valid block if found
5. Receive block reward + transaction fees

**Requirements:**

- Specialized hardware (ASICs for Bitcoin, GPUs for some altcoins)
- High electricity consumption (Bitcoin: $\approx$ 150 TWh/year globally)
- Cooling infrastructure
- High-bandwidth internet connection

**Economics:**

- Bitcoin block reward: 3.125 BTC (post-April 2024 halving)
- Break-even depends on: Hash rate, electricity cost, BTC price

**Economic incentives (block rewards + fees) align miner behavior with network security despite unreliable communication**

# How Do Validator Nodes Work in Proof-of-Stake?

**Specialized Nodes in Proof-of-Stake Blockchains**

**Functions:**

1. Stake cryptocurrency as collateral (Ethereum: 32 ETH minimum)
2. Selected to propose/attest blocks based on stake
3. Earn rewards for honest participation
4. Risk slashing (stake confiscation) for malicious behavior

**Ethereum Validator Requirements:**

- 32 ETH staked (locked)
- Run validator client 24/7 ($>$ 99% uptime)
- Consumer-grade hardware (4 GB RAM, 100 GB SSD)
- Stable internet (10 Mbps)

**Comparison to Mining:**

\+ 99.95% less energy consumption
\+ Lower hardware requirements
\- Capital locked (opportunity cost)
\- Slashing risk if offline or malicious

**Key point: Specialized Nodes in Proof-of-Stake Blockchains**

# Why Do Archive Nodes Matter?

**Full Historical State Storage**

- Store every state of the blockchain at every block height
- Enable queries like: "What was Alice's balance at block 1,000,000?"
- Required for block explorers (Etherscan, Blockchain.com)
- Needed for advanced analytics and forensics

**Storage Requirements:**

- Bitcoin archive node: $\approx$ 500 GB (same as full node, UTXO model)
- Ethereum archive node: $\approx$ 12 TB (account model stores all states)
- Grows continuously (Ethereum: $+1$ TB/year)

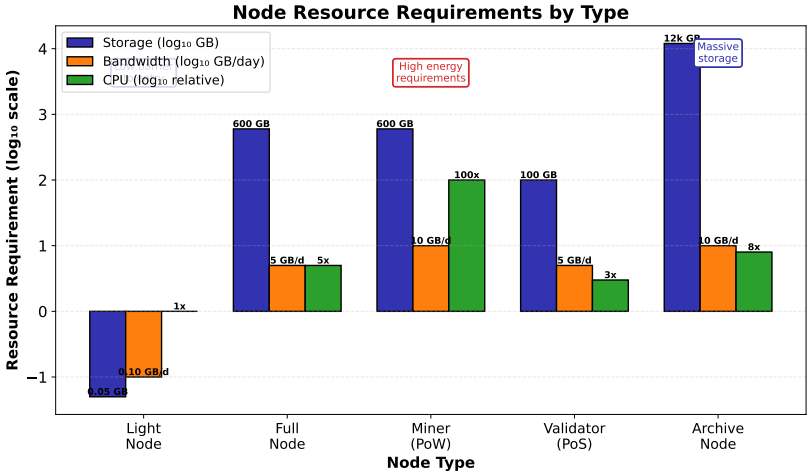
**Use Cases:**

- Block explorers
- Tax reporting services
- Academic research
- Forensic analysis

---

**Key point: Full Historical State Storage**

# How Do Node Requirements Compare?




High storage requirements create barrier to entry, reducing full nodes

**Resource requirements determine accessibility and decentralization level**

# What Makes Permissionless Blockchains Different?

## Definition

**Permissionless** (public) blockchains allow anyone to join, read, write, and participate in consensus without approval from a central authority.

**Characteristics:**

- Open participation (anyone can run a node)
- Pseudonymous identities (addresses, not real names)
- Transparent transaction history (all data public)
- Censorship resistant
- Token-based incentives (mining/staking rewards)

**Examples:** Bitcoin, Ethereum, Cardano, Solana

**Best For:** Global, trustless applications (DeFi, NFTs), censorship-resistant systems

---

**Permissionless architecture enables agreement among unknown parties by removing trusted gatekeepers at the cost of efficiency**

# Why Use Permissioned Blockchains?

## Definition

**Permissioned** blockchains restrict participation to verified entities. Access to read, write, or validate is controlled by administrators.

**Characteristics:**

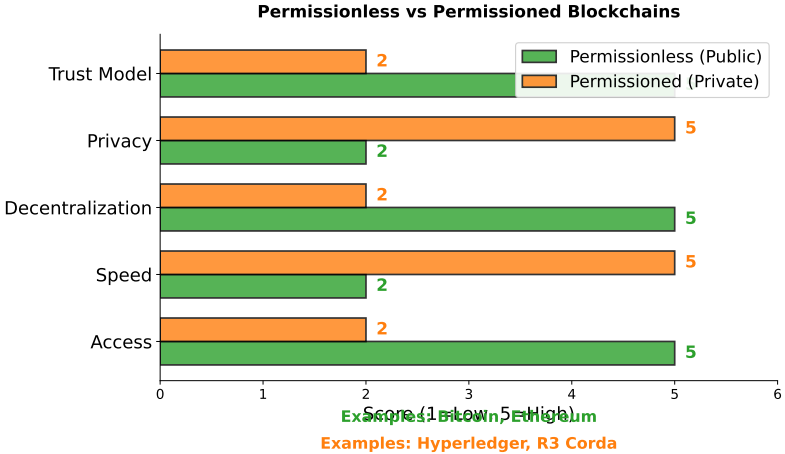
- Controlled participation (invitation-only)
- Known identities (KYC/AML compliance)
- Selective transparency (private channels possible)
- Efficient consensus (no mining, BFT-based)
- No native cryptocurrency (optional)

**Examples:** Hyperledger Fabric, R3 Corda, JP Morgan Quorum

**Best For:** Enterprise consortia (supply chain, trade finance), regulatory compliance

**Key point: Permissioned**

# How Do Permissioned and Permissionless Blockchains Compare?




**Trade-off:** Permissionless maximizes decentralization/trust; Permissioned maximizes speed/privacy.

# What Are the Trade-offs Between Permissionless and Permissioned?

**Permissionless (Public)**

**Advantages:**

- Maximum decentralization
- Censorship resistant
- Network effects (large user base)
- Innovation without permission

**Disadvantages:**

- Slower throughput (5-30 TPS)
- Higher costs (gas fees)
- Energy intensive (PoW)
- No privacy by default

**Permissioned (Private)**

**Advantages:**

- High throughput (1,000+ TPS)
- Low/no transaction fees
- Energy efficient (BFT)
- Configurable privacy

**Disadvantages:**

- Central point of control
- Limited network effects
- Potential for censorship
- Requires trust in admins

*Hybrid models exist: Public with private sidechains (e.g., Polygon for Ethereum)*

**Compare the approaches shown above**

# When Should You Choose Permissioned vs. Permissionless?

| Criterion | Permissionless | Permissioned |
|---|---|---|
| Trust Model | No trust required | Parties partially trust each other |
| Participants | Unknown, global | Known, verified entities |
| Governance | Decentralized (hard forks) | Centralized/consortium voting |
| Performance | 5-30 TPS | 1,000-10,000+ TPS |
| Privacy | Pseudonymous, transparent | Configurable, private channels |
| Use Cases | DeFi, NFTs, payments | Supply chain, trade finance, CBDCs |

**Compare the approaches shown above**

# Problem Solved? DLT Architecture

**The Original Problem**

*How do we structure data for trustless verification across thousands of nodes?*

**How DLT Architecture Solves It**

- Merkle trees enable O(log n) verification instead of downloading all transactions
- Block headers separate metadata from transaction data (80 bytes vs. MB)
- Full nodes independently validate entire history without trusting anyone

**Remaining Limitations**

- Block size limits transaction throughput (Bitcoin: 7 TPS, Ethereum: 30 TPS)
- Storage requirements create centralization pressure (600 GB+ for full nodes)

**Open Questions**

- Optimal balance between full nodes (decentralization) and light clients (accessibility)?
- Risk: As storage grows, fewer users can verify independently

*DLT architecture enables trustless verification but introduces scalability-decentralization trade-offs*

---

**Compare the approaches shown above**

# The Economics of DLT Architecture

**Incentive Structure**

- Maintaining consistent distributed state across unreliable networks
- Full nodes verify independently; light clients trust but verify headers
- Node operators pay storage ( 600GB Bitcoin) and bandwidth costs; gain sovereignty and privacy

**Economic Security**

- Independent verification ¿ trusted intermediaries
- Storage costs limit full node participation

**Key Economic Question**

**Who Pays, Who Earns?**

Node operators pay storage/bandwidth costs; users gain trustless verification and censorship resistance

**Design Principle**

$$\text{Verification Cost} \ll \text{Trust Risk}$$

**Architecture economics: Distributed state requires economic actors to bear verification costs**

**Design Space**

# Design Space: DLT Architecture

**Alternatives Considered**

1. Maximum security, 600GB storage, 2TB for Ethereum
2. Minimal storage, trust headers via SPV proofs
3. Complete historical state, 12+ TB for Ethereum

**Trade-offs Made**

- Storage vs. trustlessness
- Accessibility vs. decentralization

**Design Questions**

- How much should users trust vs verify?
- What's the UX trade-off?
- Can we reduce storage without sacrificing security?

**Key Insight**

**Storage-Trust Spectrum**

Users choose between running full nodes (maximum sovereignty) and light clients (lower barrier to entry).

**Every design is a trade-off. Understanding alternatives reveals the "why" behind choices.**

# Failure Modes: DLT Architecture

**Critical Failure Mode**

- Block propagation reaches majority before next block is mined
- Network partitions, slow propagation, malicious miners

**Root Cause**

- Assumption: Honest majority sees blocks quickly
- Violators: Eclipse attacks, network latency
- Result: Chain splits, double-spend attempts

**Historical Context**

- 2015 Bitcoin fork: BIP66 caused chain split due to uneven propagation
- 2013 Bitcoin 0.7/0.8 fork: Database limits caused temporary split

**Early Warning Signs**

- ! Eclipse attacks can isolate nodes from honest network view
- ! Increasing orphan block rate signals propagation issues
- ! Few full nodes = centralization risk

**Prediction: What could cause this to fail? How would you detect it early?**

# Key Takeaways

**What You Should Remember:**

1. **Block Structure**: Header (80 bytes) + Body (transactions), linked via cryptographic hashes
2. **Merkle Trees**: Enable efficient transaction verification ($O(\log n)$ proof size)
3. **Node Types**:
   - Full (trustless), Light (trust others)
   - Miner/Validator (consensus), Archive (historical queries)
4. **Permissioned vs. Permissionless**: Public for trustless apps, private for enterprise consortia

**Next:** L03 - Cryptographic Hash Functions

**Next Lesson:** L03 – Cryptographic Hash Functions

---

**Key point: What You Should Remember**

# Discussion Questions

**Consider and discuss:**

1. **Decentralization Trade-offs**: At what point does storage cost compromise decentralization?
2. **Node Selection**: When would you run a full node vs. use a light client?
3. **Enterprise Blockchain**: Why do enterprises often prefer permissioned chains?

---

**Key point: Consider and discuss**

Continued

# The Family Tree of Transactions

*[COMIC: Merkle trees as family genealogy]*

*"Just like you can prove your ancestry with a few certificates, you can prove a transaction belongs with just a few hashes."*

**Key Insight:** Merkle trees compress thousands of transactions into one fingerprint—change any ancestor, and the entire family tree looks different.

Thank you

Questions?

Continue to L03: Cryptographic Hash Functions

# Quiz Questions (1/2)

**Q1. What is the size of a Bitcoin block header?**
A) 32 bytes  B) 80 bytes  C) 256 bytes  D) 1 MB
**Answer: B** – Bitcoin block headers are exactly 80 bytes (6 fields).

**Q2. What is the purpose of the Merkle Root in a block header?**
A) Identify previous block  B) Prove all transactions included  C) Set mining difficulty  D) Store timestamp
**Answer: B** – Merkle Root commits to all transactions, enabling efficient verification.

**Q3. How many hashes are needed to verify a transaction in a block with 1,024 transactions using a Merkle proof?**
A) 5  B) 10  C) 512  D) 1,024
**Answer: B** – $\log_2 1024 = 10$ hashes needed for Merkle proof.

**Q4. What approximate storage is required for a Bitcoin full node?**
A) 50 MB  B) 10 GB  C) 600 GB  D) 12 TB
**Answer: C** – Bitcoin full nodes require approximately 600 GB (as of 2025).

**Q5. What does SPV stand for in the context of light nodes?**
A) Secure Payment Validation  B) Simplified Payment Verification  C) Shared Proof Verification  D) Synchronized Peer Validation
**Answer: B** – SPV (Simplified Payment Verification) described in Bitcoin whitepaper.

# Quiz Questions (2/2)

**Q6. What is the minimum stake required to run an Ethereum validator?**
A) 1 ETH B) 10 ETH C) 32 ETH D) 100 ETH
**Answer: C** – Ethereum requires exactly 32 ETH to activate a validator.

**Q7. Which node type stores every historical state at every block height?**
A) Full node B) Light node C) Archive node D) Miner node
**Answer: C** – Archive nodes store complete historical state (e.g., 12 TB for Ethereum).

**Q8. Which blockchain type typically achieves higher throughput (TPS)?**
A) Permissionless public chains B) Permissioned private chains C) Both equal D) Depends only on hardware
**Answer: B** – Permissioned chains achieve 1,000+ TPS vs. 5-30 TPS for public chains.

**Q9. What protocol do Bitcoin nodes use to propagate new blocks?**
A) HTTP B) FTP C) Gossip protocol D) TCP handshake
**Answer: C** – Gossip protocol: nodes relay to peers, who relay to their peers.

**Q10. Which field in the Bitcoin block header links blocks together?**
A) Merkle Root B) Nonce C) Previous Block Hash D) Timestamp
**Answer: C** – Previous Block Hash (32 bytes) creates the cryptographic chain.