

Lesson 14: Blocks and Cryptographic Hashing

Module 2: Blockchain and Cryptocurrencies

Digital Finance

December 12, 2025

Every Bitcoin block contains two main parts:

Block Header (80 bytes):

- **Version** (4 bytes): Protocol version
- **Previous Hash** (32 bytes): Link to parent
- **Merkle Root** (32 bytes): Transaction summary
- **Timestamp** (4 bytes): Block creation time
- **Difficulty Target** (4 bytes): Mining threshold
- **Nonce** (4 bytes): Mining puzzle solution

Block Body (Variable):

- **Transaction Counter**: Number of transactions
- **Transactions**: Actual transaction data
- Typical: 2000-3000 transactions
- Max size: 4 MB (with SegWit)
- Average: 1.5-2 MB

Block Hash:

- SHA-256 hash of the header
- Uniquely identifies block
- Must meet difficulty target

Example: Bitcoin Block 800,000

Field	Value
Block Hash	000000000000000000000000002a7c4c1e48d76c5a37902165a270156b7a8d72728a054
Previous Hash	0000000000000000000000000012117ad9f72c1c0e42329492e8c18f9e945a5d0f1b9a4
Merkle Root	7e1c6b0f5e9c8d9e3a2f1b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4
Timestamp	2023-07-13 20:42:05 UTC
Difficulty	53,911,173,001,054
Nonce	1,868,822,685
Transactions	3,285
Block Size	1,582,419 bytes (1.58 MB)
Block Reward	6.25 BTC
Total Fees	0.183 BTC
Height	800,000
Confirmations	50,000+ (as of Dec 2024)

Note: Hash starts with 19 leading zeros - probability of finding this: $1/2^{76}$

Case studies provide concrete evidence of technology impact and adoption patterns.

What is a Hash Function?

Hash Function: Mathematical algorithm that maps arbitrary data to fixed-size output

Properties:

- ① **Deterministic:** Same input always produces same output
- ② **Fast:** Quick to compute
- ③ **One-way:** Cannot reverse (pre-image resistance)
- ④ **Collision-resistant:** Hard to find two inputs with same hash
- ⑤ **Avalanche effect:** Tiny input change drastically changes output

Examples:

- MD5: 128 bits (broken, not secure)
- SHA-1: 160 bits (deprecated)
- SHA-256: 256 bits (Bitcoin)
- SHA-3: Variable (latest standard)
- BLAKE2: 256/512 bits (faster)

Output Size:

- SHA-256: 64 hex characters
- SHA-256: 2^{256} possible outputs
- More atoms in universe: $\approx 2^{265}$

Clear definitions are essential for understanding complex technical concepts.

SHA-256: The Bitcoin Hash Function

SHA-256 (Secure Hash Algorithm 256-bit) - designed by NSA, published 2001

Input/Output:

- Input: Any data (message, file, block header)
- Output: 256-bit hash (64 hexadecimal characters)
- Example: `SHA256("Hello") = 185f8db32271fe25f561a6fc938b2e264306ec304eda518007d1764826381969`

Algorithm Steps (Simplified):

- ① **Padding:** Add bits to make length $\equiv 448 \pmod{512}$
- ② **Append Length:** Add 64-bit message length
- ③ **Initialize:** Set 8 hash values (constants)
- ④ **Process:** 64 rounds of bitwise operations per 512-bit chunk
- ⑤ **Output:** Concatenate final 8 values = 256-bit hash

Security: No known practical attacks - best attack requires 2^{128} operations (infeasible)

Bitcoin remains the largest cryptocurrency by market cap and network security.

The Avalanche Effect: Demonstration

Small input change causes completely different hash:

Input	SHA-256 Hash
Hello World	a591a6d40bf420404a011733cfb7b190d62c65bf0bcd32b57b277d9ad9f146e
Hello World!	7f83b1657ff1fc53b92dc18148a1d65dfc2d4b1fa3d677284add200126d9069
hello world	b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9
Hello Worle	44896a8f3e6ec4f93c8d3c92d72c8df3e3c74b5d6e9e8f3a5b6c7d8e9f0a1b2c

Analysis:

- Adding single character (!) changes 100% of hash bits
- Changing case (H to h) changes entire hash
- Changing one letter (d to e) produces unrelated hash
- No pattern or correlation between similar inputs

Implication: Cannot predict hash output - must compute to verify

Key concepts from this slide inform practical applications in finance.

SHA-256 produces 2^{256} possible hashes - how big is that?

- $2^{256} \approx 1.16 \times 10^{77}$ different hashes
- Estimated atoms in observable universe: 10^{80}
- Estimated grains of sand on Earth: 10^{24}
- Age of universe in seconds: 4.3×10^{17}

Birthday Paradox and Collision Probability:

- To find collision with 50% probability: Need to hash 2^{128} inputs
- At 1 trillion hashes/second: Would take 10^{25} years
- For comparison: Sun will die in 5×10^9 years

Practical Conclusion: SHA-256 collisions are computationally infeasible

Cryptographic primitives provide the security foundation for blockchain systems.

Pre-image Resistance: The One-Way Property

Given a hash, can you find the original input?

Forward (Easy):

$$\text{Input} \xrightarrow{\text{SHA-256}} \text{Hash}$$

Reverse (Impossible):

$$\text{Hash} \xrightarrow{???} \text{Input}$$

- Instant computation
- Example: $\text{SHA256("Bitcoin") = b4056df6...}$
- Deterministic and fast

- No mathematical inverse
- Only option: Brute force
- Try all possible inputs
- Computationally infeasible

Example Attack: Find input for hash 000000000019d6689c085ae165831e93...

- Average attempts needed: 2^{255} (half the hash space)
- At 1 quadrillion attempts/second: 10^{61} years
- Even with all computers on Earth: Still infeasible

Key concepts from this slide inform practical applications in finance.

Hash Pointers: Linking Blocks Together

Hash Pointer: Data structure that contains both location AND cryptographic hash of data

Regular Pointer:

- Points to memory address
- Can retrieve data
- No integrity check
- Data can be tampered

Example:

- Pointer: 0x7ffe5367e044
- Data at address: "Alice pays Bob 5 BTC"
- If data changes: No detection

Hash Pointer:

- Points to data location
- Contains hash of data
- Can retrieve AND verify
- Tamper-evident

Example:

- Location: Block 100
- Hash: 00000000000080b66c911bd5ba14a74db...
- If data changes: Hash mismatch detected

Blockchain Application: Each block header contains hash pointer to previous block

Cryptographic primitives provide the security foundation for blockchain systems.

Why Blockchain is Immutable

Changing a single transaction breaks the entire chain:

Scenario: Attacker tries to modify transaction in Block 100

- ① Modify transaction data in Block 100
- ② Merkle root changes (transaction hash changed)
- ③ Block 100 hash changes (header data changed)
- ④ Block 101's "Previous Hash" field no longer matches
- ⑤ Block 101 becomes invalid
- ⑥ Must recalculate Block 101 hash (requires solving PoW puzzle)
- ⑦ Block 102's "Previous Hash" no longer matches
- ⑧ Must recalculate Block 102, 103, 104... all subsequent blocks
- ⑨ While attacker recalculates, honest miners add new blocks
- ⑩ Attacker must outpace entire network (51% attack)

Conclusion: Older blocks are exponentially harder to modify - after 6 confirmations (1 hour), considered practically irreversible

AI and ML are transforming financial services through automation and prediction.

Merkle Tree: Binary tree of hashes allowing efficient verification of large datasets

Structure (Bottom-up):

- ① **Leaf Nodes:** Hash of each transaction
- ② **Parent Nodes:** Hash of concatenated child hashes
- ③ **Root:** Final hash (Merkle root) in block header

Benefits:

- Compact representation of all transactions
- Verify transaction inclusion with $O(\log n)$ hashes
- SPV (Simplified Payment Verification) for light clients
- Only need block headers + Merkle proof

Example (8 transactions):

- Full verification: Download all 8 transactions
- Merkle proof: Download 3 hashes only
- Proof size: $\log_2(8) = 3$ hashes
- For 2000 transactions: Only 11 hashes needed

SPV Use Case: Mobile Bitcoin wallet verifies payment without downloading full blockchain (500+ GB)

Key concepts from this slide inform practical applications in finance.

Merkle Tree Example: 4 Transactions

Constructing Merkle Root:

Transactions: $T_1 = \text{"Alice to Bob 1 BTC"}$
 $T_2 = \text{"Carol to Dave 2 BTC"}$
 $T_3 = \text{"Eve to Frank 0.5 BTC"}$
 $T_4 = \text{"Grace to Heidi 3 BTC"}$

Step 1 - Leaf Hashes:

$$\begin{aligned}H_1 &= \text{SHA256}(T_1) = \text{a3c8...} \\H_2 &= \text{SHA256}(T_2) = \text{7f4b...} \\H_3 &= \text{SHA256}(T_3) = \text{9e2d...} \\H_4 &= \text{SHA256}(T_4) = \text{1c5a...}\end{aligned}$$

Step 2 - Parent Hashes:

$$\begin{aligned}H_{12} &= \text{SHA256}(H_1 \| H_2) = \text{d8f3...} \\H_{34} &= \text{SHA256}(H_3 \| H_4) = \text{6b9e...}\end{aligned}$$

Step 3 - Merkle Root:

$$\text{MerkleRoot} = \text{SHA256}(H_{12} \| H_{34}) = \text{4e7a...}$$

Case studies provide concrete evidence of technology impact and adoption patterns.

Merkle Proof: Verifying Transaction Inclusion

Problem: Light client wants to verify T_2 is in block, without downloading all transactions

Solution: Full node provides Merkle proof (3 hashes):

- H_1 (sibling of T_2)
- H_{34} (sibling of H_{12})
- Merkle root from block header

Verification Steps:

- ① Client has T_2 and computes $H_2 = \text{SHA256}(T_2)$
- ② Receives H_1 from proof, computes $H_{12} = \text{SHA256}(H_1||H_2)$
- ③ Receives H_{34} from proof, computes MerkleRoot = $\text{SHA256}(H_{12}||H_{34})$
- ④ Compares computed Merkle root with block header
- ⑤ If match: Transaction confirmed in block

Efficiency: For n transactions, proof size = $O(\log_2 n)$ hashes

- 1,000 transactions: 10 hashes (320 bytes)
- 1,000,000 transactions: 20 hashes (640 bytes)

Key concepts from this slide inform practical applications in finance.

Difficulty Target and Leading Zeros

Bitcoin Mining Puzzle: Find nonce such that block hash \mid target

Target Representation:

- Difficulty target: Maximum allowed hash value
- Higher difficulty = Lower target = More leading zeros required
- Current (Dec 2024): 19 leading zeros

Leading Zeros	Probability	Average Attempts
1	1/16	16
4	1/65,536	65,536
8	$1/4.3 \times 10^9$	4.3 billion
16	$1/1.8 \times 10^{19}$	18 quintillion
19	$1/1.5 \times 10^{23}$	150 sextillion

Example Valid Hash: 000000000000000000000002a7c4c1e48d76c5a37902165a270156b7a8d72728a054

Key concepts from this slide inform practical applications in finance.

Difficulty Adjustment Mechanism

Goal: Maintain 10 minute average block time despite changing network hash rate

Algorithm:

- Every 2,016 blocks (2 weeks)
- Calculate actual time taken vs. expected time (20,160 minutes)
- Adjust difficulty: $\text{NewDifficulty} = \text{OldDifficulty} \times \frac{\text{Expected Time}}{\text{Actual Time}}$
- Maximum adjustment: 4x increase or 1/4 decrease per period

Example (2023):

- Period: Blocks 780,000 - 782,015
- Expected: 20,160 minutes (14 days)
- Actual: 18,900 minutes (13.125 days) - blocks came faster
- Adjustment: Difficulty increased by 6.3%

Historical Trend:

- 2009: Difficulty = 1 (CPU mining)
- 2024: Difficulty = 73 trillion (ASIC mining)
- 10^{13} times harder than genesis block

Key concepts from this slide inform practical applications in finance.

Hash Rate: Network Computing Power

Hash Rate: Total number of hash computations per second across network

Period	Hash Rate	Mining Technology
2009	5 MH/s	CPU (Satoshi solo mining)
2010	200 GH/s	GPU mining begins
2013	20 TH/s	ASIC mining begins
2017	10 EH/s	Industrial mining farms
2021	180 EH/s	Peak before China ban
2024	600 EH/s	Post-halving recovery

Scale:

- $600 \text{ EH/s} = 600,000,000,000,000,000,000$ hashes per second
- More computing power than top 500 supercomputers combined
- Each hash attempt: $\approx 1/600$ quintillion chance of success

Network metrics provide objective measures of adoption and ecosystem health.

Block Propagation and Orphan Blocks

Problem: Network latency - two miners find valid blocks simultaneously

Scenario:

- ① Miner A finds Block 700,000 at 12:00:00
- ② Miner B finds different Block 700,000 at 12:00:01
- ③ Both blocks are valid and propagate through network
- ④ Network temporarily splits (some nodes see A, others see B)
- ⑤ Miners start working on both competing chains
- ⑥ Miner C finds Block 700,001 building on Block A
- ⑦ Longest chain rule: Block A chain wins
- ⑧ Block B becomes “orphan block” - discarded
- ⑨ Miner B loses 6.25 BTC reward

Statistics:

- Orphan rate: 0.5% of blocks (rare but happens)
- Average propagation time: 1-2 seconds globally
- Why 10 minutes?: Minimize orphans while maintaining decentralization

Key concepts from this slide inform practical applications in finance.

Blockchain Data Structures Summary

Structure	Purpose	Properties
Hash Function	Fingerprint data	Deterministic, one-way, collision-resistant, avalanche effect
Hash Pointer	Link + verify	Points to data + contains hash for integrity
Merkle Tree	Efficient verification	$O(\log n)$ proof size, enables SPV
Block Header	Minimal representation	80 bytes contains all block metadata
Blockchain	Tamper-evident ledger	Chain of hash pointers, immutable
Difficulty Target	Mining puzzle	Adjusts to maintain 10-min block time

Key Insight: All blockchain security stems from cryptographic hash functions - if SHA-256 breaks, Bitcoin breaks

Security Analysis: What Could Go Wrong?

Hash Function Attacks:

- **Pre-image:** Find input from hash
- **Collision:** Find two inputs with same hash
- **Length extension:** Exploit hash construction

SHA-256 Status:

- **Pre-image:** 2^{256} security (safe)
- **Collision:** 2^{128} security (safe)
- **Length extension:** Not applicable to Bitcoin (double hashing)

Quantum Computing Threat:

- Grover's algorithm: $\sqrt{2^{256}} = 2^{128}$ speedup
- Still computationally infeasible
- Bigger threat: Public key crypto (covered next lesson)

Mitigation Strategies:

- SHA-3 ready as backup
- Post-quantum hash functions (SPHINCS+)
- Bitcoin can hard fork if needed

Current Assessment: No practical threat to SHA-256 in foreseeable future

Practical Exercise: Computing Block Hash

Simplified Example: Calculate hash for mini-block

Block Header (simplified):

- Previous Hash: 00000000000000000001
- Merkle Root: abcdef123456
- Timestamp: 1702000000
- Nonce: ?

Task: Find nonce such that hash starts with 4 zeros (0000...)

Brute Force Approach:

- ① Concatenate header: 00000000000000000001|abcdef123456|1702000000|0
- ② Compute SHA-256: 7a3b2c1d... (no leading zeros)
- ③ Increment nonce: Try 1, then 2, then 3...
- ④ Keep trying until hash starts with 0000
- ⑤ Expected attempts: $2^{16} = 65,536$

Solution:Nonce 45,892 produces 0000f7a3b2c1d...

Cryptographic primitives provide the security foundation for blockchain systems.

Block Finality: How Many Confirmations?

Confirmations: Number of blocks built on top of transaction's block

Confirmations	Time	Security Level
0	0 min	Unconfirmed - in mempool only, easily reversible
1	10 min	Low - vulnerable to accidental orphaning
3	30 min	Medium - coffee purchase acceptable
6	60 min	High - exchanges require this, 99.9% final
100	17 hours	Very High - coinbase maturity (mining reward)

Attack Cost Analysis (6 confirmations):

- Attacker must mine 7 blocks faster than honest network
- Requires 51% hash rate for extended period
- Cost: \$500,000/hour in electricity + equipment
- Only profitable for large transaction values

Rule of Thumb: Value \leq \$1,000: 1-3 confirmations; Value \geq \$10,000: 6+ confirmations

Understanding the process flow is key to identifying optimization opportunities.

Lesson 15: Public Key Cryptography and Signatures

What We'll Cover:

- Symmetric vs asymmetric encryption
- Elliptic Curve Cryptography (ECC) fundamentals
- Public/private key pairs in Bitcoin
- Digital signatures (ECDSA)
- Address generation and wallet structure
- Security best practices

Prepare:

- Review basic modular arithmetic ($a \bmod n$)
- Understand concept of mathematical groups
- Install Bitcoin wallet for hands-on key generation

- ① **Block Structure:** 80-byte header + variable transaction body
- ② **SHA-256:** One-way, collision-resistant hash with 2^{256} output space
- ③ **Avalanche Effect:** Tiny input change completely alters hash
- ④ **Hash Pointers:** Enable tamper-evident data structures
- ⑤ **Merkle Trees:** Efficient verification with $O(\log n)$ proof size
- ⑥ **Immutability:** Changing old block requires recalculating all subsequent blocks
- ⑦ **Difficulty:** Adjusts every 2016 blocks to maintain 10-min average
- ⑧ **Finality:** 6 confirmations (1 hour) considered irreversible

"Blockchain security is reducible to hash function security - SHA-256 is the foundation."