BLOCKCHAIN: SMART CONTRACTS LECTURE 7 - **BITCOIN-PART2**

FLORIN CRACIUN

IMPORTANT

Some of the following slides are the property of

Dr. Emanuel Onica & Dr. Andrei Arusoaie

Faculty of Computer Science,

Alexandru Ioan Cuza University of Iași

and are used with their consent.

RECALL FROM PREVIOUS LECTURE

- Addresses & Keys
- Wallets
- Transactions

THIS LECTURE

- Network
- Blocks
  - Merkle trees
  - Bloom filters
- Mining

THE BITCOIN NETWORK

- Peer-to-peer architecture
  - Nodes interconnect in a mesh net with a “flat” topology
  - No server, no centralized service, no hierarchy
  - Resilient
- Bitcoin network = nodes running the bitcoin P2P protocol

- Node functions:
  - route, validate and propagate transactions and blocks,
  - discover and maintain connections to peers

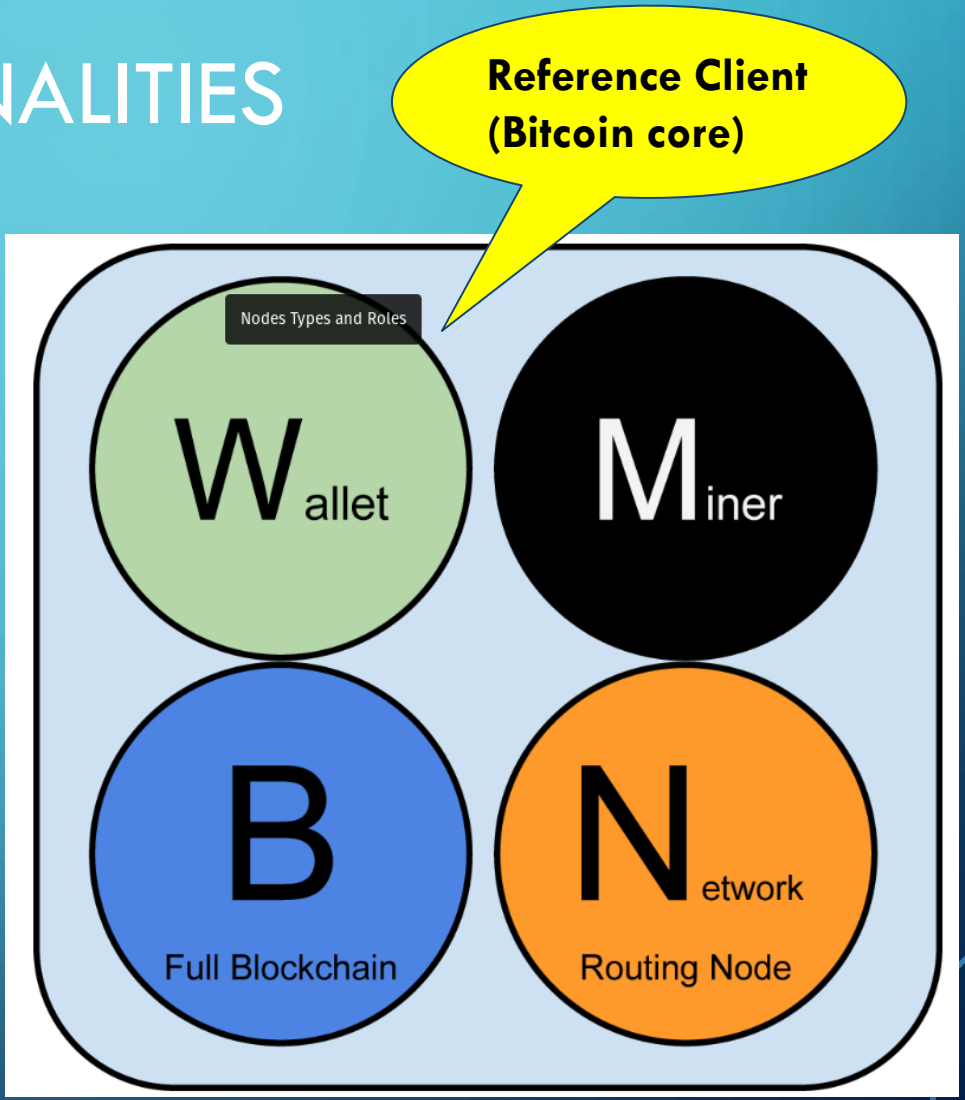
NODE FUNCTIONALITIES

- Wallet
- Miner
- Full Blockchain
- Network routing node

- Gateway Routing Servers:

Pool server

Stratum server



Source: Mastering Bitcoin, Andreas M. Antonopoulos

FULL BLOCKCHAIN NODES

- Hold a complete up-to-date copy of the blockchain
- They can verify any transaction without external reference
- They are **not** lightweight nodes
- +Gb persistent storage
- Complete independence
- They build a full database of UTXO

SPV

- Nodes that maintain only a subset of the blockchain
- They verify transactions using SPV: *simplified payment verification*
- An SPV node cannot validate whether the UTXO is unspent like full nodes can do
  - Uses Merkle paths to link the transaction (UTXO) to its block B1
  - Wait until 6 blocks have been piled on top of B1
  - By proxy: the UTXO is not a double-spend
- Vulnerabilities:
  - Hide transactions from SPV nodes; SPV is not aware about all transactions
  - Denial-of-service, double-spending attacks

MINING NODES AND WALLETS

- Mining nodes compete for creating new blocks
  - Use specialized hardware for proof-of-work
  - Sometimes, pool servers are used for lightweight nodes
  - Solo Miner: B + M + N
  - Thin miners: M + **S**, M + **P**
- Wallets:
  - typically part of a full node when desktop clients are used
  - Constrained devices (smartphones) = SPV nodes
  - Lightweight wallets: W + **S**, W + N

NETWORK DISCOVERY

- How to (re)discover nodes on the network?
  - *Trusted* Bitcoin nodes
  - Existing Bitcoin nodes are selected at random
  - Use TCP, port 8333
    - "Handshake"
- How does a new node find peers?
  - "Seed nodes" = long-running stable nodes
  - Connect to seed nodes and broadcast its own IP to neighbours
  - After rebooting it is important to remember the most successful connections

ALERT MESSAGES

- Alert messages are a seldom used function
- ``Emergency broadcast system''
- Implemented in most nodes
- They allow core devs to send emergency messages to all nodes
- The alert system has only been used a handful of times, most notably in early 2013 when a critical database bug caused a multiblock fork to occur in the bitcoin blockchain. (https://bitcoin.org/en/alert/2013-03-11-chain-fork)

BIBLIOGRAPHY

Further reading about the Bitcoin network:

- *Mastering Bitcoin*, Andreas M. Antonopoulos
  - Chapter 6: https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch06.html

BLOCKS IN BITCOIN

- Block
  - Header
    - 80 bytes
    - Version number
    - Previous block hash
    - Merkle root: hash of the root of the transactions' merkle tree
    - Difficulty target
    - Nonce
  - Transactions
    - Average transaction: 250 bytes
    - https://charts.bitcoin.com/btc/chart/transactions-per-block#5ma4

BLOCK STRUCTURE

Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields form the block header
1-9 bytes (VarInt)	Transaction Counter	How many transactions follow
Variable	Transactions	The transactions recorded in this block

Source: Mastering Bitcoin, Andreas M. Antonopoulos

BLOCK HEADER

Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Difficulty Target	The proof-of-work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the proof-of-work algorithm

Source: Mastering Bitcoin, Andreas M. Antonopoulos

HOW BLOCKS ARE IDENTIFIED

- Block header Hash
  - Apply SHA256 twice to the block
  - It identifies the block uniquely and unambiguously
  - It is computed by each node independently
- Block height
  - Another way to identify a block
  - https://live.blockcypher.com/btc/

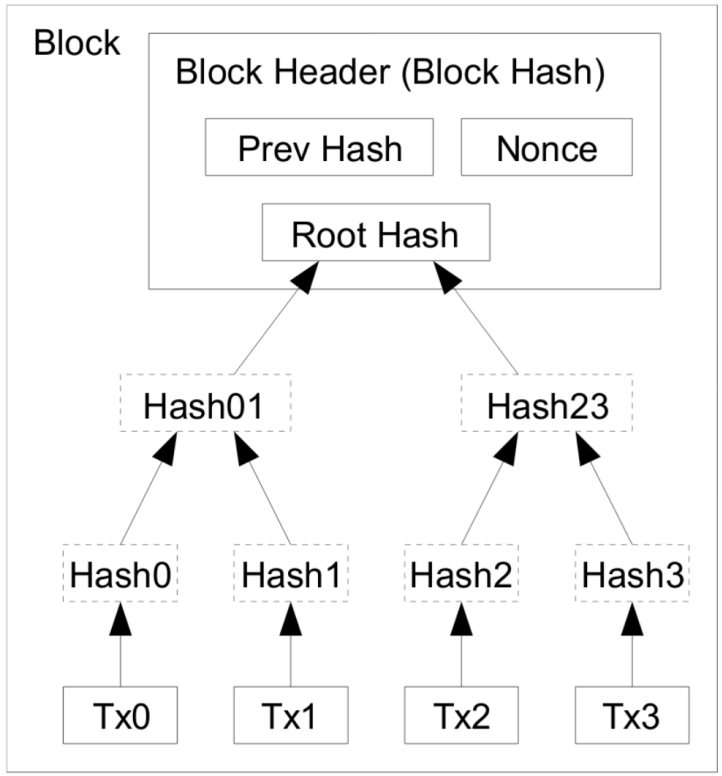
THE GENESIS BLOCK

- The first block in the Bitcoin's blockchain
- Created in 2009
- Genesis block hash:
  - `000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f`
  - `Height = 0`
    - https://www.blockchain.com/btc/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

LINKING BLOCKS

- Nodes receive blocks from the network
- They discover the previous block by looking in the previous block hash field in the header

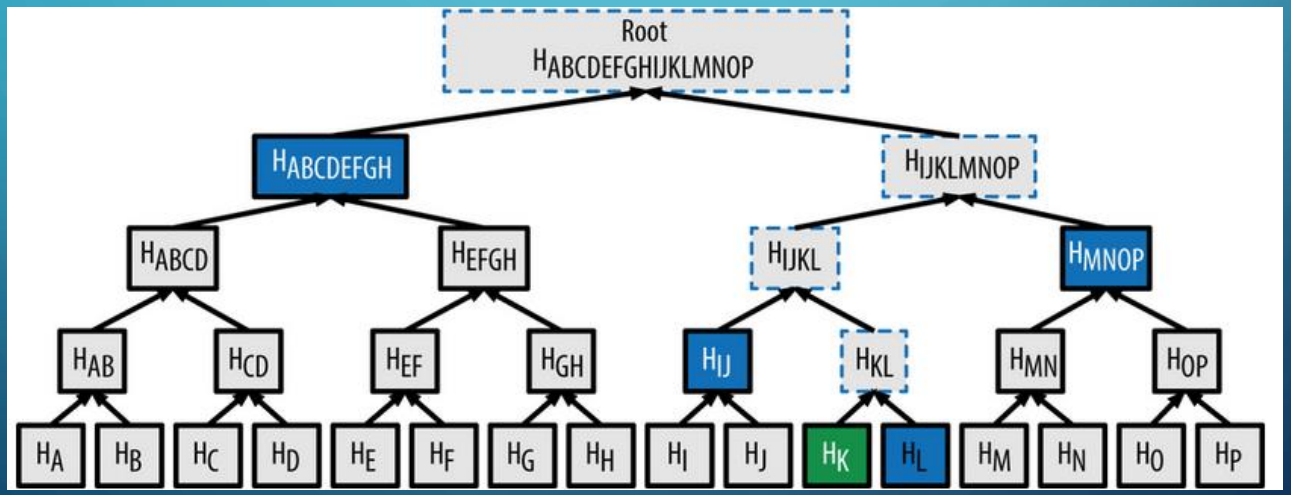
MERKLE TREES



Source: Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto, Link: https://bitcoin.org/bitcoin.pdf

MERKLE PATH

- **Authentication path** or **Merkle path**
  - Proves that a specific transaction is included in a block
  - 2 * log(n) time spent to check ih Hk included in block



MERKLE TREE EFFICIENCY

- The amount of data that needs to be exchanged as a merkle path to prove that a transaction is part of a block

Source: Mastering Bitcoin, Andreas M. Antonopoulos

Number of transactions	Approx. size of block	Path size (hashes)	Path size (bytes)
16 transactions	4 kilobytes	4 hashes	128 bytes
512 transactions	128 kilobytes	9 hashes	288 bytes
2048 transactions	512 kilobytes	11 hashes	352 bytes
65,535 transactions	16 megabytes	16 hashes	512 bytes

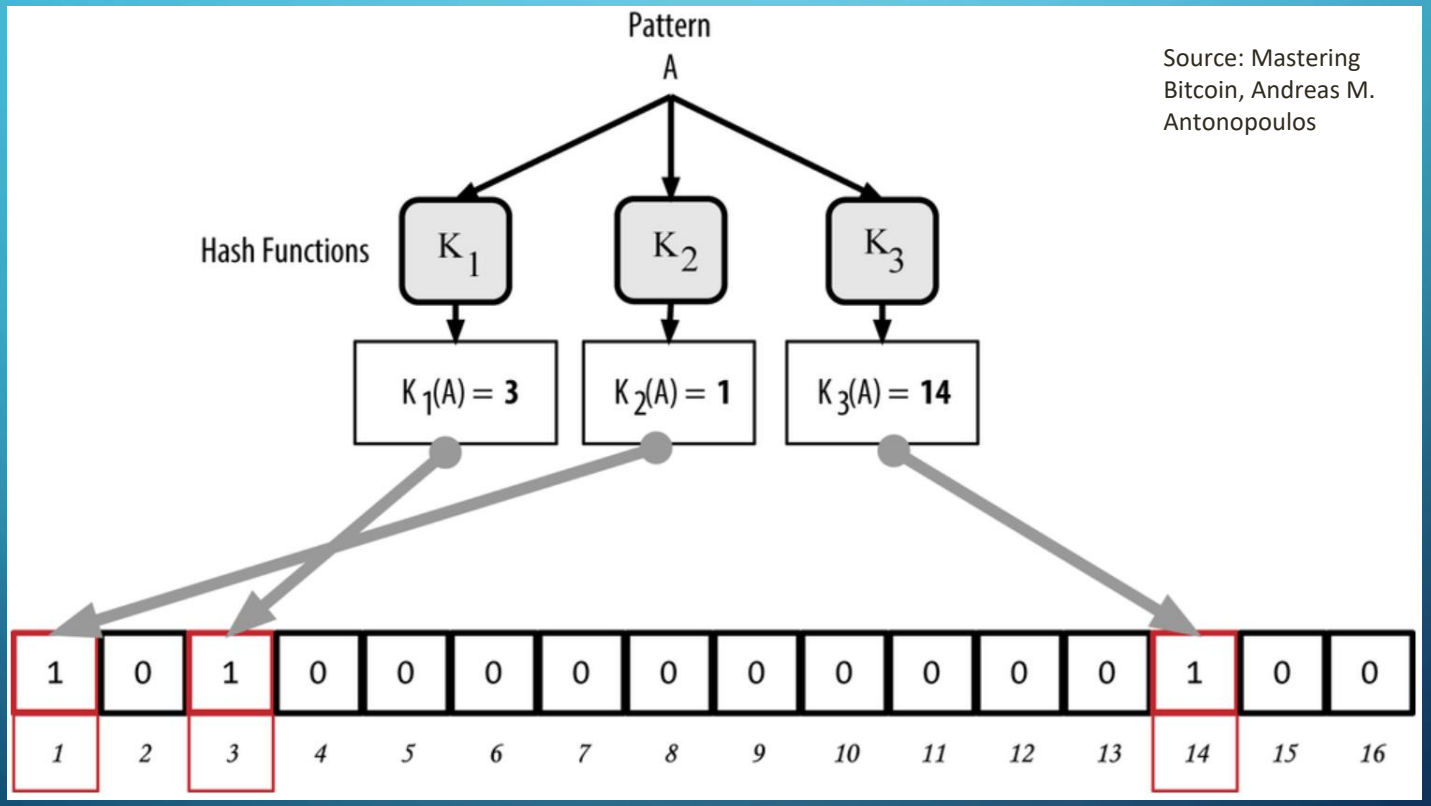
SPV & MERKLE TREES

- SPV nodes: limited devices (e.g., smartphones)
  - SPVs don't keep all transactions
  - SPVs don't download all the blocks
- SPVs use Merkle paths to validate transactions
  - SPVs ask their connections for a merkle path
  - Bloom filters are used to specify a specific pattern

BLOOM FILTERS

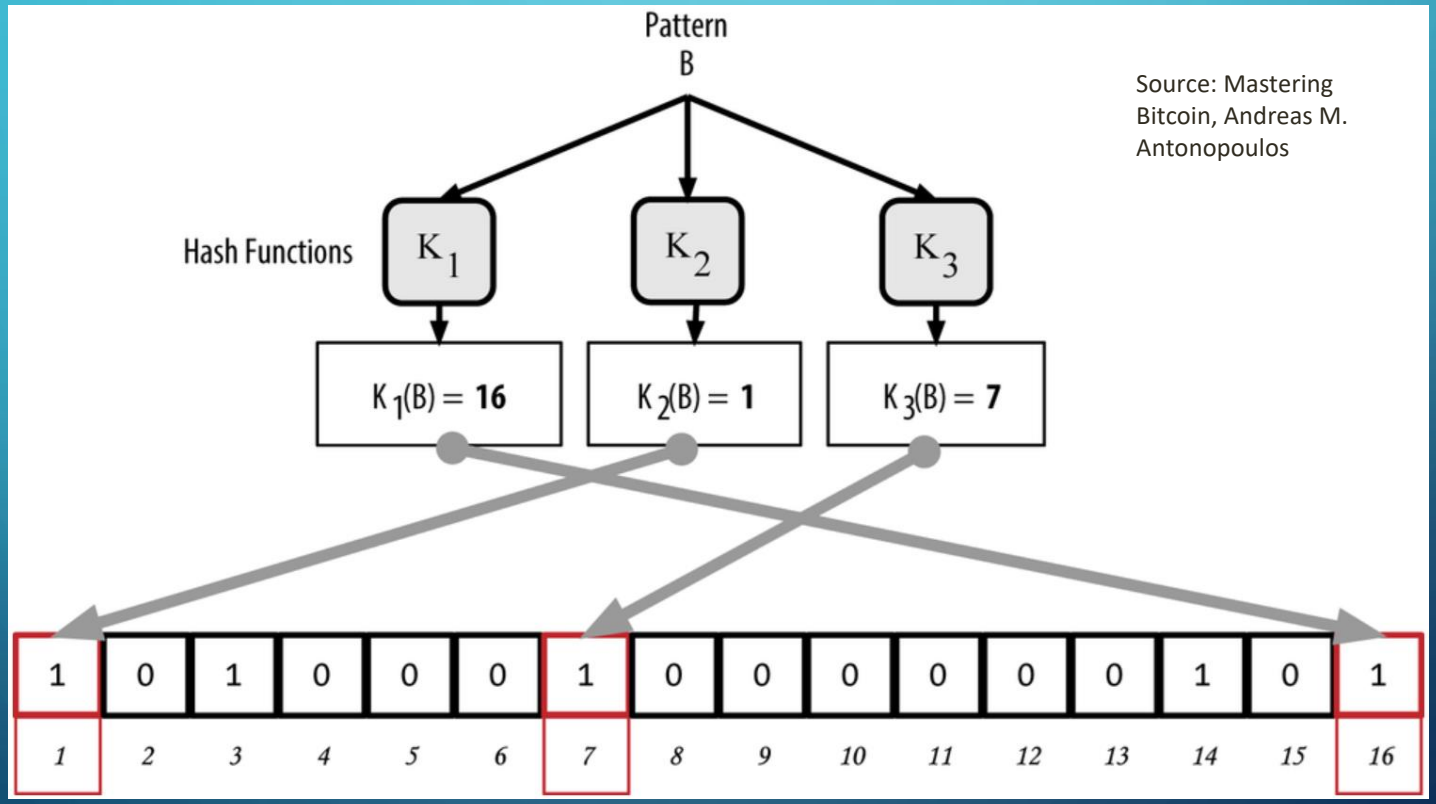
- Used to specify a search pattern for transactions
- Consist of:
  - An array of N-binary digits (initially, all digits are 0)
  - A number of M hash functions that generate (integer) values between 1 and N

ADDING A VALUE TO THE FILTER



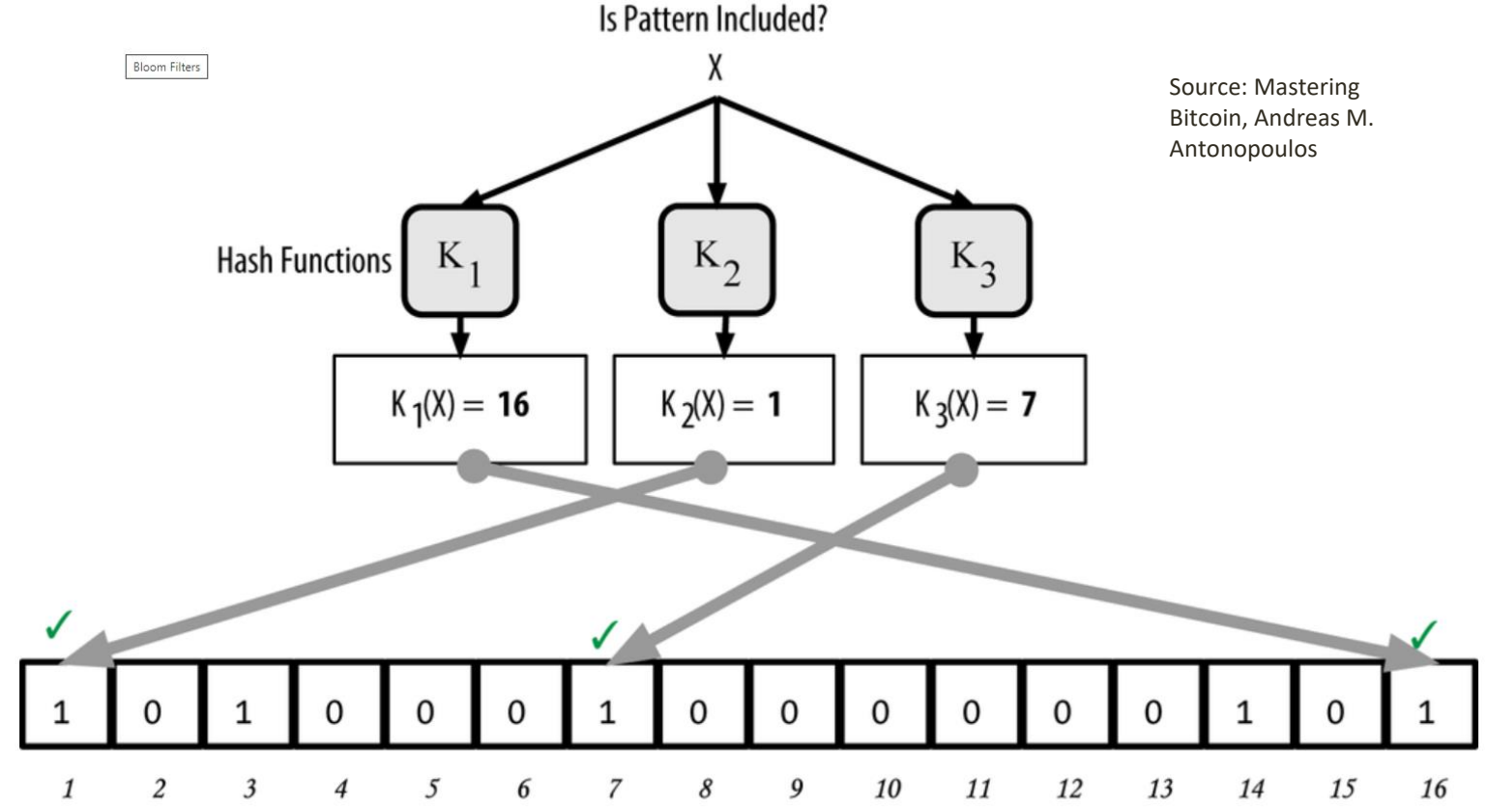
Source: Mastering Bitcoin, Andreas M. Antonopoulos

ADDING B VALUE TO THE FILTER



Source: Mastering Bitcoin, Andreas M. Antonopoulos

VERIFICATION



Source: Mastering Bitcoin, Andreas M. Antonopoulos

BIBLIOGRAPHY

Further reading about the Bitcoin network:

- *Mastering Bitcoin*, Andreas M. Antonopoulos
  - Chapter 7: https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch07.html

MINING - INTRO

- Mining
  - The process of creating new blocks
  - Secures the bitcoin system against fraudulent transactions
  - New bitcoin is added to the money supply
- A new block is mined in about 10 minutes
- Transactions registered in the blockchain are confirmed by the miners

MINING AND FEES

- Two sources of rewards for miners:
  - Mining fees associated to blocks
  - Transaction fees collected from the transactions included in the mined block
- Mining = solve a difficult mathematical problem based on a cryptographic hash algorithm
- The solution of this difficult problem = proof that the miner expended significant computing effort (proof-of-work)
- Mining fee: halved each 210000 blocks
- At some point (2140) no new bitcoin will be issued

ECONOMICS - BRIEF

- For the first 4 years, 50 BTC/block were created
- Nov 2012: 25BTC/block
- 2016: 12.5BTC/block
- 2020, May; 6.25 BTC/block
- Max no of halvings: 64 → ETA 2140
  - No new BTC will be added to the network
- Almost 21 million BTC, will be issued in total
- Deflation? Scenarios?

DECENTRALIZED CONSENSUS

- How do nodes agree on which is the "real" blockchain?
- The blockchain is assembled independently by each node in the network
- *Emergent consensus* - consensus is not achieved explicitly
  - emergent artifact of the asynchronous interaction of thousands of independent nodes, all following simple rules

DECENTRALIZED CONSENSUS - STAGES

1. Independent verification of each transaction, by every full node, based on a comprehensive list of criteria
2. Independent aggregation of those transactions into new blocks by mining nodes, coupled with demonstrated computation through a proof-of-work algorithm
3. Independent verification of the new blocks by every node and assembly into a chain
4. Independent selection, by every node, of the chain with the most cumulative computation demonstrated through proof of work

INDEPENDENT VERIFICATION OF TRANSACTIONS

- New transactions: references to UTXOs + unlocking script + new tx output with locking scripts
- New transactions are propagated across the network
- Every node checks if the received transaction is valid
  - Only valid transactions are propagated in the network
  - Invalid transactions are discarded at the first node that encounters them
- There is a long list of checking criteria

TRANSACTION CHECKING CRITERIA

1. The transaction's syntax and data structure must be correct.
2. Neither lists of inputs or outputs are empty.
3. The transaction size in bytes is less than MAX_BLOCK_SIZE.
4. Each output value, as well as the total, must be within the allowed range of values (less than 21m coins, more than 0).
5. None of the inputs have hash=0, N=–1 (coinbase transactions should not be relayed).

TRANSACTION CHECKING CRITERIA

6. nLockTime is less than or equal to INT_MAX.
7. The transaction size in bytes is greater than or equal to 100.
8. The number of signature operations contained in the transaction is less than the signature operation limit.
9. The unlocking script (scriptSig) can only push numbers on the stack, and the locking script (scriptPubkey) must match isStandard forms (this rejects “nonstandard” transactions).

TRANSACTION CHECKING CRITERIA

10. A matching transaction in the pool, or in a block in the main branch, must exist.
11. For each input, if the referenced output exists in any other transaction in the pool, the transaction must be rejected.
12. For each input, look in the main branch and the transaction pool to find the referenced output transaction. If the output transaction is missing for any input, this will be an orphan transaction. Add to the orphan transactions pool, if a matching transaction is not already in the pool.

TRANSACTION CHECKING CRITERIA

13. For each input, if the referenced output transaction is a coinbase output, it must have at least COINBASE_MATURITY (100) confirmations.
14. For each input, the referenced output must exist and cannot already be spent.
15. Using the referenced output transactions to get input values, check that each input value, as well as the sum, are in the allowed range of values (less than 21m coins, more than 0).
16. Reject if the sum of input values is less than sum of output values.
17. The unlocking scripts for each input must validate against the corresponding output locking scripts.

TRANSACTION CHECKING CRITERIA

- These conditions change over time
  - Check https://github.com/bitcoin/bitcoin

MINING NODES

- … or *miners*
- Tasks of a miner:
  - Listens for new blocks
  - Listens for new transactions and adds them to a *transaction pool*
  - Validates transactions
  - Aggregates transactions into blocks
  - Competes for block generation by performing proof-of-work

MINING PREPARATION - I

- A mining node keeps a local copy of the blockchain
- Picks transactions from the transaction pool and attempts to build a *candidate block*:
  - First, it applies a priority metric:
    - Priority = Sum (Value of inputs * Input Age) / Transaction Size
    - Input age = is the number of blocks that have elapsed since the UTXO was recorded on the blockchain
    - Transaction Size = no of bytes
  - The first 50 kilobytes of transaction space in a block are set aside for high-priority transactions.
  - Older transactions will have a higher priority

MINING PREPARATION - II

- Continued:
  - The node can now fill the rest of the block up to the maximum size of the block
    - Prioritize the transactions with the highest fee
    - Transaction fee = Sum(inputs) - Sum(outputs)
  - The first transaction is the *coinbase transaction*
    - Payment to the current node wallet: mining fee
    - This special transaction has no references to previous UTXOs

NORMAL TRANSACTION VS. COINBASE TRANSACTION

Size	Field	Description
32 bytes	Transaction Hash	Pointer to the transaction containing the UTXO to be spent
4 bytes	Output Index	The index number of the UTXO to be spent, first one is 0
1-9 bytes (VarInt)	Unlocking-Script Size	Unlocking-Script length in bytes, to follow
Variable	Unlocking-Script	A script that fulfills the conditions of the UTXO locking script.
4 bytes	Sequence Number	Currently disabled Tx-replacement feature, set to 0xFFFFFFFF

VS.

Size	Field	Description
32 bytes	Transaction Hash	All bits are zero: Not a transaction hash reference
4 bytes	Output Index	All bits are ones: 0xFFFFFFFF
1-9 bytes (VarInt)	Coinbase Data Size	Length of the coinbase data, from 2 to 100 bytes
Variable	Coinbase Data	Arbitrary data used for extra nonce and mining tags in v2 blocks, must begin with block height
4 bytes	Sequence Number	Set to 0xFFFFFFFF

MINING PREPARATION - III : CONSTRUCT THE BLOCK HEADER

- The mining node needs to fill in six fields, as listed in this table:

Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Difficulty Target	The proof-of-work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the proof-of-work algorithm

MINING : PROOF OF WORK

- Use SHA256 function to compute a hash of the block until the obtained hash is less than a certain value (difficulty)
  - The mining node does that by modifying the nonce field
  - The difficulty is updated once every two weeks
    - By decreasing the value -> it gets more difficult to find the hash

New Difficulty = Old Difficulty * (Actual Time of Last 2016 Blocks / 20160 mins)

This formula enforces an average of 10 min per block

SUCCESS!

- If the proof-of-work finished before receiving a new block B from other nodes then
  - The mining node transmits B to all its peers
  - The other nodes receive, validate, and add B to their own copy of the blockchain
  - Moreover, the other nodes *abandon* their efforts to find a block at the same height, and they start to compute another block on top of B

BLOCK VALIDATION

1. The block data structure is syntactically valid
2. The block header hash is less than the target difficulty (enforces the proof of work)
3. The block timestamp is less than two hours in the future (allowing for time errors)
4. The block size is within acceptable limits
5. The first transaction (and only the first) is a coinbase generation transaction
6. All transactions within the block are valid using the transaction checklist discussed in Independent Verification of Transactions

ASSEMBLY CHAINS OF BLOCKS

- assembly of blocks into chains
- select the chain with the most proof of work!
  - The “longest” chain
- Nodes maintain three sets of blocks:
  - those connected to the main blockchain
  - those that form branches off the main blockchain (secondary chains)
    - Reconverge on the secondary chain
  - blocks that do not have a known parent in the known chains (orphans).
    - Network issues, inconsistencies, etc.

BLOCKCHAIN FORKS

- They can happen
- Two new blocks B1 and B2 arrive at the “same” time
  - Mining nodes keep them both
  - However, when a new block C arrives, it will have as parent either B1 or B2
- Forks are almost always resolved within one block

MINING POOLS

- Miners now collaborate to form mining pools
- They pool their hashing power and share the reward among thousands of participants
- By participating in a pool, miners get a smaller share of the overall reward, but typically get rewarded every day!
- Pool operator = the owner of the pool server
  - He gets a fee as well…
  - *Managed* mining pool

CONSENSUS ATTACKS

- Bitcoin is vulnerable to attack by miners (or pools)
  - attempt to use their hashing power to dishonest or destructive ends
- The consensus mechanism depends on having a majority of the miners acting honestly out of self-interest
- Attacks can affect the most recent blocks and cause denial-of-service
- 51% attack

BIBLIOGRAPHY

Further reading about the Bitcoin network:

- *Mastering Bitcoin*, Andreas M. Antonopoulos
  - Chapter 8: https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch08.html

SUMMARY

- Network
  - P2P, decentralized network, nodes and roles
  - Network discovery
- Blocks
  - Structure
  - Genesis block
  - Merkle paths and Bloom filters
- Mining
  - Transactions in blocks, coinbase transaction
  - Priorities, validations (tx and blocks), consensus