# Mastering Bitcoin 2nd edition

<https://github.com/bitcoinbook/bitcoinbook/blob/develop/book.asciidocb>

## Bitcoin consists of:

* A decentralized peer-to-peer network (the bitcoin protocol)
* A public transaction ledger (the blockchain)
* A set of rules for independent transaction validation and currency issuance (consensus rules)
* A mechanism for reaching global decentralized consensus on the valid blockchain (Proof-of-Work algorithm)

## Three basic questions for anyone accepting digital money are:

* Can I trust that the money is authentic and not counterfeit?
* Can I trust that the digital money can only be spent once (known as the “double-spend” problem)?
* Can I be sure that no one else can claim this money belongs to them and not me?

## Keys:

* secp256k1 key pair

## Address:

* RIPEMD160( SHA256( public\_key ))
* Encoded in Base58Check:
  + Base64 without 0, O, l, I, +, / to avoid typo
  + Prefix + data + checksum
  + 4 bytes of checksum at the end: first 4 bytes of sha256(sha256(prefix+data))

## Wallet:

* Bitcoin wallets contain keys, not coins
* nondeterministic wallet:
  + Each key is independently generated from a random number
  + Discouraged, too cumbersome to back up and use
* deterministic wallet
  + All the keys are derived from a single random master key (seed)
  + the seed is sufficient to recover all the derived keys, and therefore a single backup at creation time is sufficient
  + BIP39: Generate mnemonicwords for user to record, can have passphrase, HMAC-SHA512, 256 bits used as private key and 256 bits used as chain code withchild key derivation function togenerate child keys. Can be an infinite tree structure.

## Transaction:

* Transaction outputs consist of two parts:
  + An amount of bitcoin, denominated in *satoshis*, the smallest bitcoin unit
  + A cryptographic puzzle that determines the conditions required to spend the output. Also known as locking script, witness script or scriptPubKey
* Transaction Inputs:
  + A transaction ID, referencing the transaction that contains the UTXO being spent
  + An output index (vout), identifying which UTXO from that transaction is referenced (first one is zero)
  + A scriptSig, which satisfies the conditions placed on the UTXO, unlocking it for spending
  + A sequence number (to be discussed later)
* Transaction Fees
  + Rewards for miner
  + Making it economically infeasible for attackers to flood the network with transactions.
  + Calculated based on the size of the transaction in kilobytes
  + Prioritised
  + Transaction fees are implied: Fees = Sum(Inputs) – Sum(Outputs)
* Transaction Scripts and Script Language
  + Turing Incompleteness: scripts have limited complexity and predictable execution times, no infinite loop or logic bomb
* Stateless Verification
* locking script: specifies the conditions that must be met to spend the output in the future
* unlocking script: satisfies, the conditions placed on an output by a locking script and allows the output to be spent
* From left to right, uses stack to process:
  + 2 3 OP\_ADD 5 OP\_EQUAL : 2 + 3 == 5
  + Transactions are valid if the top result on the stack is TRUE
* Separate execution of unlocking and locking scripts
  + concatenated and executed in sequence has a vulnerability that allowed a malformed unlocking script to push data onto the stack and corrupt the locking script
  + unlocking script is executed with a result, then locking script is execute with the result copied from unlocking script. Final result is true then unlocking script successfully resolving the condition
* Sample:
  + OP\_DUP OP\_HASH160 <Cafe Public Key Hash> OP\_EQUALVERIFY OP\_CHECKSIG
  + <Cafe Signature> <Cafe Public Key>
  + Execution: Duplicate public key, hash it, compare with the value and then check signature
* The Importance of Randomness in Signatures
  + If the same value k is used in the signing algorithm on two different transactions, the private key can be calculated and exposed to the world!
  + use RFC 6979 or a similarly deterministic-random algorithm to ensure you generate a different k for each transaction
* Multisignature scripts
  + N public keys are recorded in the script and at least M of those must provide signatures to unlock the funds
  + Locking script: M <Public Key 1> <Public Key 2> ... <Public Key N> N CHECKMULTISIG
  + Unlocking script: 0 <Signature B> <Signature C>
    - 0 has to be added due to a bug which pops 1 extra item
* Pay-to-Script-Hash (P2SH)
  + locking script that is replaced by a hash is referred to as the *redeem script* because it is presented to the system at redemption time rather than as a locking script
  + not able to put a P2SH inside a P2SH redeem script
  + redeem script is not presented to the network until you attempt to spend a P2SH output. Might accidentally lock bitcoin in such a way that it cannot later be spent.
* Data Recording Output (RETURN)
  + Someone uses bitcoin to record data hashes. Causes the output can never be spent and increases the UTXO set size
  + RETURN operator allows developers to add 80 bytes of nonpayment data to a transaction output. Recorded in blockchain but not UTXO set
  + Format: RETURN <data>
  + At most 80 bytes
  + no "unlocking script" that corresponds to RETURN
* Timelocks
  + Timelocks are restrictions on transactions or outputs that only allow spending after a point in time
  + Check Lock Time Verify (CLTV)
    - <now + 3 months> CHECKLOCKTIMEVERIFY DROP DUP HASH160 <Bob's Public Key Hash> EQUALVERIFY CHECKSIG
  + Relative Timelocks (CSV)
    - specify, as a condition of spending an output, an elapsed time from the confirmation of the output in the blockchain
    - CHECKSEQUENCEVERIFY
  + Median-Time-Past
    - calculated by taking the timestamps of the last 11 blocks and finding the median
  + Timelock Defense Against Fee Sniping
    - a theoretical attack scenario, where miners attempting to rewrite past blocks "snipe" higher-fee transactions from future blocks to maximize their profitability
    - when Bitcoin Core creates transactions, it uses nLocktime to limit them to the "next block," by default
* Scripts with Flow Control (Conditional Clauses)
  + Using the IF, ELSE, ENDIF, and NOTIF opcodes.
  + Can contain boolean operators such as BOOLAND, BOOLOR, and NOT.
  + condition being evaluated comes before the IF opcode because of stack
* Segregated Witness
  + the term "witness" is used to describe a solution to a cryptographic puzzle
  + an architectural change to bitcoin that aims to move the witness data from the scriptSig (unlocking script) field of a transaction into a separate witness data structure that accompanies a transaction
  + Why? Transaction Malleability, Script Versioning, Network and Storage Scaling, Signature Verification Optimization, Offline Signing Improvement
  + Backward compatible

## Bitcoin Network

* Peer-to-Peer Network Architecture
* Node Types and Roles
  + Four functions: wallet, miner, full blockchain database, and network routing
* Extended Bitcoin Network: Nodes running other protocols can connect through gateway routers
* Bitcoin Relay Networks
  + a network that attempts to minimize the latency in the transmission of blocks between minersb
* Network Discovery
  + Usual port 8333
  + DNS servers provide a list of IP addresses of bitcoin nodes
  + Handshake version, addr, getaddr
* Full Nodes
* Simplified Payment Verification (SPV) Nodes
  + download only the block headers
  + verifies transactions by reference to their depth in the blockchain, establish a link between the transaction and the block that contains it, using a merkle path
  + Privacy issue: keep track of all the transactions requested by a wallet on an SPV node and use those to associate bitcoin addresses with the user of that wallet
  + Bloom filter:
    - a probabilistic search filter, a way to describe a desired pattern without specifying it exactly
    - Use M number of hash functions to map data to a number N bit array. If some data maps to all 1s in the array, it may fit. If any 0, it must not fit
* Encrypted and Authenticated Connections
* Transaction Pools
  + a temporary list of unconfirmed transactions called the memory pool, mempool, or transaction pool
* UTXO pool only contains confirmed outputs

## Blockchain

* Structure of a block:
  + Block size (4 bytes), Block header (80 bytes), Transaction counter (1 – 9 bytes), Transactions (x bytes)
* Block header
  + Version (4 bytes)
  + Previous block hash (32 bytes)
  + Merkle root (32 bytes)
  + Timestamp (4 bytes)
  + Difficulty target (4 bytes)
  + Nonce (4 bytes)
* Block Identifiers: Block Header Hash and Block Height
  + Block header hash:
    - hashing the block header twice through the SHA256
    - always identify a single block uniquely. (If new block hash is the same as old one, it will be discarded)
  + Block height:
    - position in the blockchain, first is 0
    - two or more blocks might compete for a single position in the blockchain
* Genesis Block
  + common ancestor of all the blocks in the blockchain
  + hard coded in app as the trust anchor
* Merkle Trees
  + binary hash tree, is a data structure used for efficiently summarizing and verifying the integrity of large sets of data
  + constructed by recursively hashing pairs of nodes until there is only one hash, called the root, or Merkle root
  + If there is an odd number of transactions to summarize, the last transaction hash will be duplicated to create an even number of leaf nodes, also known as a balanced tree
  + authentication path or Merkle path connecting the specific transaction to the root of the tree
* Merkle Trees and Simplified Payment Verification (SPV)
  + Merkleblock
* Bitcoin’s Test Blockchains
  + A blockchain for testing purposes

## Mining and Consensus

* Mining is the mechanism by which bitcoin’s *security* is *decentralized*.
* Miners receive two rewards: new coins created, and transaction fees from all the transactions in the block
* miners compete to solve a difficult mathematical problem based on a cryptographic hash algorithm. The solution to the problem, called the Proof-of-Work, is included in the new block and acts as proof that the miner expended significant computing effort
* Deflationary Money
* Decentralized Consensus
* Bitcoin’s decentralized consensus emerges from the interplay of four processes that occur independently on nodes across the network:
  + Independent verification of each transaction, by every full node, based on a comprehensive list of criteria
  + Independent aggregation of those transactions into new blocks by mining nodes, coupled with demonstrated computation through a Proof-of-Work algorithm
  + Independent verification of the new blocks by every node and assembly into a chain
  + Independent selection, by every node, of the chain with the most cumulative computation demonstrated through Proof-of-Work
* The Coinbase Transaction: The first transaction in any block is a special transaction, called a *coinbase transaction*
  + No input. Contains Coinbase Reward and Fees paying to miner self
  + No unlocking script but coinbase data
* Constructing the Block Header
* Mining the Block
* Proof-of-Work Algorithm
  + produce a hash that is less than the target
  + Retargeting to Adjust Difficulty. Every 2,016 blocks, all nodes retarget the Proof-of-Work
* Successfully Mining the Block
* Validating a New Block
* Assembling and Selecting Chains of Blocks
* Blockchain Forks
* Mining and the Hashing Race
* The Extra Nonce Solution
  + a miner could find a block by iterating through the nonce until the resulting hash was below the target
  + Timestamp change and coinbase transaction data change as nonce to guarantee
* Mining Pools
  + individual miners working alone don’t stand a chance
  + measure the individual contributions: Proof-of-Work algorithm with a lower difficulty to win the share of bitcoin rewards
  + Managed pools
    - Managed by pool operator
    - Cheated centralized pool operator may direct the pool effort to double-spend transactions or invalidate blocks
  + Peer-to-peer mining pool (P2Pool):
    - a peer-to-peer mining pool without a central operator
    - decentralizing the functions of the pool server, implementing a parallel blockchain-like system called a share chain
* Consensus Attacks
  + vulnerable to attack by miners (or pools) that attempt to use their hashing power to dishonest or destructive ends
  + double-spend attack: either before a transaction is confirmed, or if the attacker takes advantage of a blockchain fork to undo several blocks
    - a merchant selling large-value items must wait at least six confirmations before giving the product to the buyer
    - alternatively, the merchant should use an escrow multi-signature account, again waiting for several confirmations after the escrow account is funded
  + denial-of-service: deny service to specific bitcoin participants (specific bitcoin addresses)

51% attack scenario doesn’t actually require 51% of the hashing power

* + - Security research groups have used statistical modeling to claim that various types of consensus attacks are possible with as little as 30% of the hashing power
* Changing the Consensus Rules
  + Hard Forks: may diverge into following two chains
  + Soft Forks: A soft fork is a forward-compatible change to the consensus rules that allows unupgraded clients to continue to operate in consensus with the new rules
    - can only be used to constrain the consensus rules, not to expand them
    - Bitcoin Script had ten opcodes reserved for future use, NOP1 through NOP10, interpreted as a null-potent operator, meaning they have no effect
    - Criticisms of Soft Forks: Technical debt, Validation relaxation, Irreversible upgrades
    - 降低安全性，新的协议在旧的节点上没有经过验证就写入，可能导致某些不合法在旧的算力上被写入