

Lesson 18: Bitcoin Architecture

Module 2: Blockchain Fundamentals

Digital Finance

`charts/lesson_18/bitcoin_system_architecture.pdf`

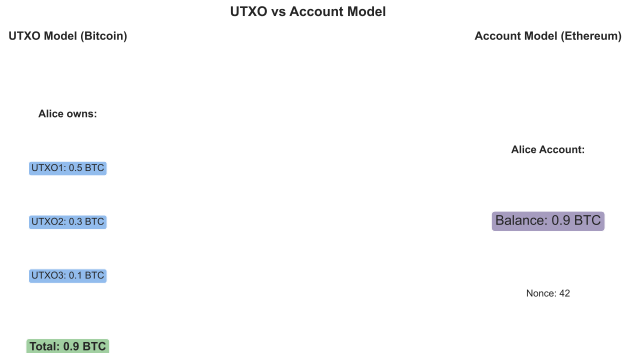
Physical Cash:

- Discrete bills and coins
- Cannot split a \$20 bill
- Give exact amount or get change
- Destroyed when spent

UTXO Model:

- Discrete chunks of bitcoin
- Cannot partially spend UTXO
- Consume entirely, create change UTXO
- Marked as spent, new UTXO created

charts/lesson_18/utxo_concept.pdf



UTXO (Bitcoin):

- Stateless
- Better privacy (addresses change)
- Parallel transaction validation
- No account balances

Account (Ethereum):

- Stateful (balance stored)
- Simple mental model
- Sequential nonces
- Easier for smart contracts

Transaction Structure: Inputs and Outputs

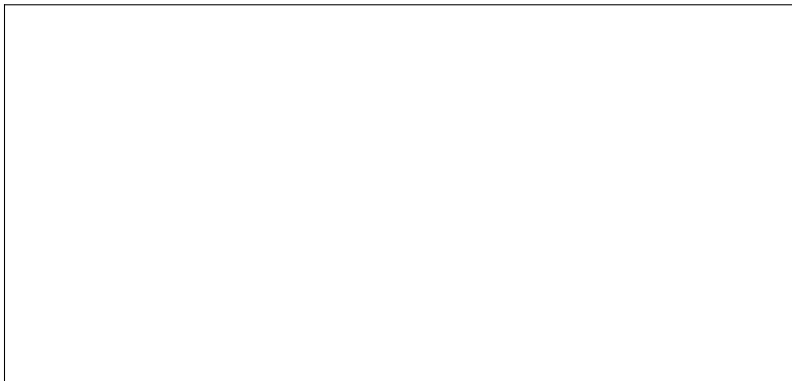
`charts/lesson_18/transaction_structure.pdf`

Example Transaction: Alice Pays Bob

Scenario: Alice has 0.5 BTC UTXO, wants to pay Bob 0.3 BTC

Transaction:

- **Input:** Alice's 0.5 BTC UTXO + Alice's signature
- **Output 1:** 0.3 BTC to Bob's address
- **Output 2:** 0.19 BTC change to Alice's new address
- **Fee:** 0.01 BTC ($0.5 - 0.3 - 0.19$)



`charts/lesson_18/transaction_chain.pdf`

What is Script?

- Stack-based language
- Turing-incomplete (no loops)
- Defines spending conditions
- Executed by all nodes

Script Types:

- Pay-to-Public-Key-Hash (P2PKH)
- Pay-to-Script-Hash (P2SH)
- Pay-to-Witness-Public-Key-Hash (P2WPKH)
- Multisig

`charts/lesson_18/script_execution.pdf`

Locking Script (scriptPubKey):

```
OP_DUP OP_HASH160 <PubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

Unlocking Script (scriptSig):

```
<Signature> <PublicKey>
```

Execution:

- 1 Push signature and public key to stack
- 2 Duplicate public key
- 3 Hash duplicated public key
- 4 Compare hash to stored hash (EQUALVERIFY)
- 5 Verify signature with public key (CHECKSIG)
- 6 Success if top of stack is TRUE

Multisig: M-of-N Signatures

Use Case:

- Shared custody
- Corporate accounts
- Escrow services
- Enhanced security

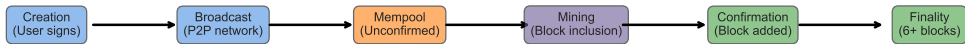
Example (2-of-3):

- 3 public keys
- Require any 2 signatures
- Escrow: buyer, seller, arbiter

[charts/lesson_18/multisig_example.pdf](#)

Script: `OP_2 <PubKey1> <PubKey2> <PubKey3> OP_3 OP_CHECKMULTISIG`

Bitcoin Transaction Lifecycle



Stages:

- 1 **Creation:** User signs transaction

Properties:

- Unconfirmed transactions
- Not consensus-critical (each node maintains own)
- Size varies by network congestion
- Transactions ranked by fee rate

Fee Market:

- Higher fee = faster inclusion
- Fee rate: satoshis per byte (sat/vB)
- Dynamic pricing based on demand

[charts/lesson_18/mempool_visualization.pdf](#)

Fee Calculation:

$$\text{Fee} = \sum \text{Inputs} - \sum \text{Outputs}$$

Fee Rate:

$$\text{Fee Rate} = \frac{\text{Fee}}{\text{Transaction Size (vB)}}$$

Example:

- Transaction size: 250 vB
- Target: 50 sat/vB
- Fee: $250 \times 50 = 12,500 \text{ sat} = 0.000125 \text{ BTC}$

[charts/lesson_18/fee_distribution.pdf](#)

`charts/lesson_18/fee_market_congestion.pdf`

Problem:

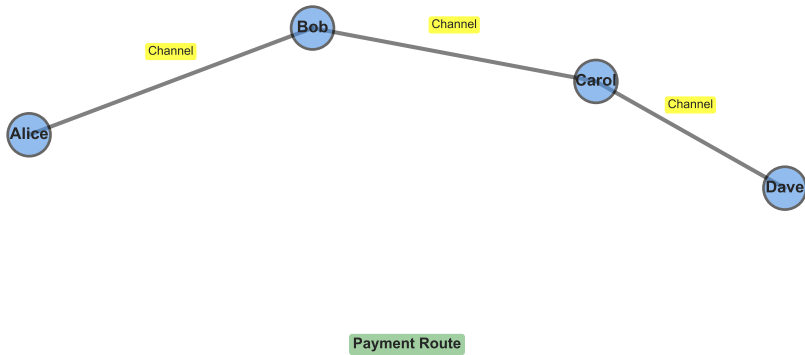
- Block size limit: 1 MB
- Signature data takes 60–70% of transaction
- Scalability bottleneck
- Transaction malleability

Solution:

- Separate signature data (witness)
- Witness data not counted toward 1 MB limit
- Effective block size: ~2–2.5 MB
- Fixes malleability

charts/lesson_18/segwit_structure.pdf

Lightning Network: Off-Chain Payment Channels



Problem: Bitcoin throughput ~ 7 tx/s, expensive fees, slow confirmations

Opening Channel:

- ① Alice and Bob create 2-of-2 multisig
- ② Deposit funds (funding transaction)
- ③ On-chain confirmation

Off-Chain Payments:

- Exchange signed commitment transactions
- Update balances instantly
- No on-chain transactions
- Unlimited throughput

[charts/lesson_18/lightning_channel.pdf](#)

`charts/lesson_18/lightning_routing.pdf`

Aspect	On-Chain (Layer 1)	Lightning (Layer 2)
Speed	10 min average	Instant (milliseconds)
Throughput	~7 tx/s	Millions of tx/s
Fees	\$1–\$50 (variable)	<\$0.01
Finality	6 confirmations (1 hour)	Instant (with channel counterparty)
Trust	Fully trustless	Counterparty risk (can close unilaterally)
Use Case	Large settlements, savings	Micropayments, retail

Improvements:

- Schnorr signatures (batch verification)
- MAST (Merkalized Alternative Script Trees)
- Enhanced privacy (multisig looks like single-sig)
- More efficient smart contracts

Benefits:

- Lower fees for complex scripts
- Better privacy
- Enables new use cases (DLCs, multi-party protocols)

[charts/lesson_18/taproot_comparison.pdf](#)

Responsibilities:

- Download entire blockchain
- Validate all transactions
- Relay transactions and blocks
- Enforce consensus rules

Requirements (2024):

- Disk: ~600 GB (growing ~50 GB/year)
- Bandwidth: ~500 GB/month
- RAM: 4+ GB
- CPU: Modest (validation not mining)

`charts/lesson_18/node_types.pdf`

- **UTXO Model:** Discrete outputs, like digital cash, stateless validation
- **Transactions:** Inputs (previous UTXOs) + Outputs (new UTXOs) + Fee
- **Script:** Stack-based language defines spending conditions (P2PKH, multisig)
- **Mempool:** Unconfirmed transactions, fee market determines priority
- **SegWit:** Separated witness data, increased capacity, fixed malleability
- **Lightning:** Layer 2, instant payments, millions tx/s, minimal fees

Next Lesson: Ethereum and Smart Contracts – from currency to computation