Bitcoin Protocol Deep Dive

BSc Blockchain, Crypto Economy & NFTs

Course Instructor

Module A: Blockchain Foundations

Learning Objectives

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

- Explain the UTXO (Unspent Transaction Output) model
- Describe the structure of a Bitcoin transaction
- Describe transaction inputs and outputs
- Recognize different Bitcoin Script types
- Trace the lifecycle of a transaction from creation to confirmation
- Distinguish between legacy and SegWit transaction formats

Building on L05: Public Key Cryptography

UTXO Confusion

[COMIC: A confused person at a cash register trying to pay $25, but their wallet contains only $20 and $10 bills – no $25 bill exists. The cashier explains: "You give me $30, I give you $5 change." Caption: "Bitcoin doesn't have balances – just unspent coins."]

[Comic placeholder – to be illustrated]

- Unlike bank accounts, Bitcoin tracks individual "coins" (UTXOs), not balances
- Spending requires consuming entire UTXOs and creating change outputs

Key point: UTXO Confusion

The Problem: How do we prevent digital counterfeiting?

The Challenge
How do we prevent digital counterfeiting and double-spending without a central authority tracking who owns what? Digital files can be copied infinitely at zero cost – how can digital money have scarcity?

Why It Matters

- Every pre-Bitcoin digital currency required trusted intermediaries who could be shut down, hacked, or corrupted
- Historical examples: DigiCash (centralized issuer, bankrupt 1998), E-gold (central database, seized by FBI 2007), Liberty Reserve (prosecuted for money laundering 2013)

What We Need

- Trustless verification mechanism
- Unforgeable, publicly verifiable transaction history where anyone can verify ownership without trusting a third party

The Cryptoeconomics Question
Coordinating without trusted intermediaries

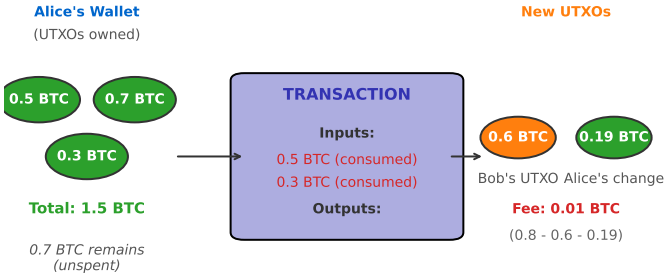
Today's lesson: How Bitcoin's transaction protocol addresses this challenge

How Does the UTXO Model Track Ownership?

What is a UTXO?

- Unspent Transaction Output = a chunk of bitcoin that can be spent
- Bitcoin does not track account balances (unlike Ethereum)
- Instead, tracks individual "coins" (UTXOs)

UTXO Model: Unspent Transaction Outputs



UTXOs are consumed entirely, change returned as new UTXO

UTXOs create unforgeable ownership – each coin spent only once with valid signature

How Are UTXOs Like Physical Cash?

Analogy: Physical Cash

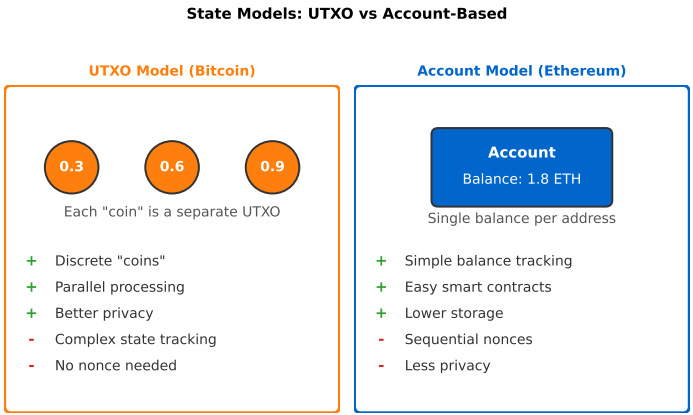
- UTXOs = bills in your wallet
- No "balance" – just individual coins
- Pay 30 EUR: use 20 + 10 bills
- Pay 25 EUR with 50: get 25 change

Key Principles:

- Each UTXO spent once (consumed)
- Spending creates new UTXOs
- Blockchain tracks unspent UTXOs
- Balance = sum of your UTXOs

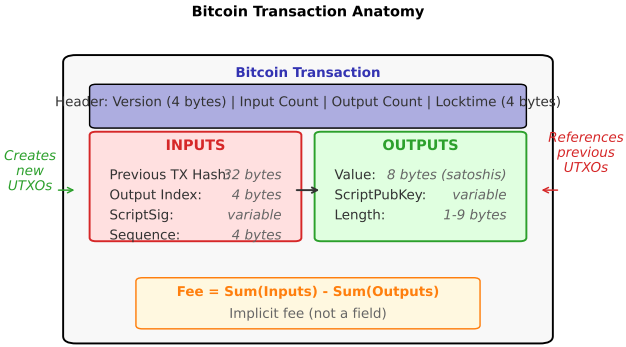
Compare the approaches shown above

Why Did Bitcoin Choose UTXOs Over Accounts?



Why Bitcoin Uses UTXO: Easier to verify (check UTXO existence), no global state needed, natural double-spend prevention.

How is a Bitcoin Transaction Structured?



Transaction Hash (txid):

- Double SHA-256 of entire transaction – unique identifier
- Used to reference transaction in inputs

Key point: Transaction Hash (txid)

What Information Do Transaction Inputs Contain?

Each Input Contains:

1. **Previous TX Hash:** txid of UTXO
2. **Output Index:** which output (0,1,2...)
3. **ScriptSig:** signature + pubkey
4. **Sequence Number:** for replacement

Example Input:

- Prev tx: `5a3c7b...` (3 BTC)
- Output index: 0
- ScriptSig: ownership proof

Multiple Inputs:
Combine UTXOs; each signed separately

→ **Problem: How do we prevent digital... — Transaction Inputs — Every input must prove ownership via digital signature, preventing unauthorized spending**

How Do Transaction Outputs Work?

Each Output Contains:

1. **Value:** satoshis (1 BTC = 10^8 sat)
2. **ScriptPubKey:** spending conditions

Change Outputs:

- Input ¿ payment $\rightarrow$ create change
- Change to sender (new address)
- 5 BTC $\rightarrow$ 3 + 1.999 change

Transaction Fee:

- Fee = Inputs − Outputs
- Implicit (not stated)
- Miner collects difference

Compare the approaches shown above

How Does Bitcoin Script Verify Transactions?

What is Bitcoin Script?

- Simple, stack-based programming language
- Not Turing-complete (no loops, limited expressiveness)
- Executed during transaction validation
- Determines whether transaction is valid

How It Works:

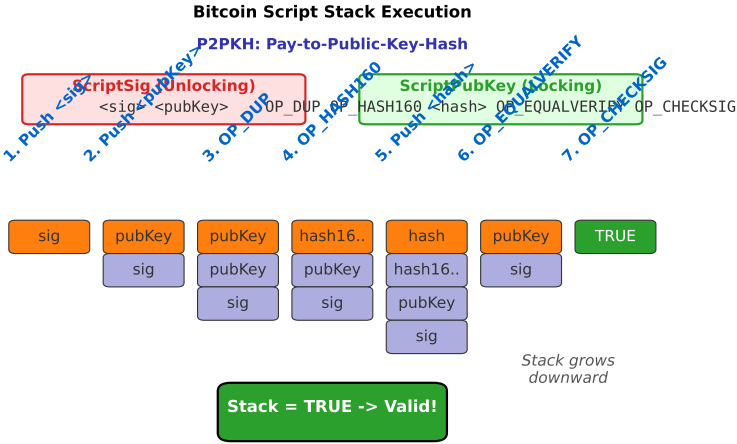
1. Combine ScriptSig (from input) + ScriptPubKey (from previous output)
2. Execute script operations left to right
3. Use a stack (LIFO data structure)
4. Transaction valid if final stack value is TRUE

Basic Operations:

- `OP_DUP`: duplicate top stack item
- `OP_HASH160`: hash with SHA-256 then RIPEMD-160
- `OP_EQUALVERIFY`: check equality, fail if not
- `OP_CHECKSIG`: verify signature against public key

Key point: What is Bitcoin Script?

How Does P2PKH Script Execution Work?



Why Public Key Hash? Shorter addresses (20 vs 33 bytes), extra security (quantum resistance until spending).

How Does P2PKH Script Execution Work? – visual summary

How Does Pay-to-Script-Hash Enable Complex Transactions?

Purpose:

- Allows complex spending conditions (multi-signature, time-locks)
- Hides complexity until spending time
- Sender only needs recipient's P2SH address

ScriptPubKey:

```
OP_HASH160 <ScriptHash> OP_EQUAL
```

ScriptSig:

```
<Signature1> <Signature2> ... <RedeemScript>
```

Verification Process:

1. Hash the redeem script
2. Verify hash matches ScriptHash in ScriptPubKey
3. Execute redeem script with provided signatures
4. Transaction valid if redeem script evaluates to TRUE

Study this code pattern carefully

Problem Check-In: Mining Economics

Recall Our Problem

How do we prevent digital counterfeiting?

What We've Learned So Far

- UTXO model tracks coin ownership cryptographically – each output is spent exactly once
- Bitcoin Script verifies spending conditions without trusted intermediaries
- Together, these prevent counterfeiting: no valid signature = no spending

Still to Address

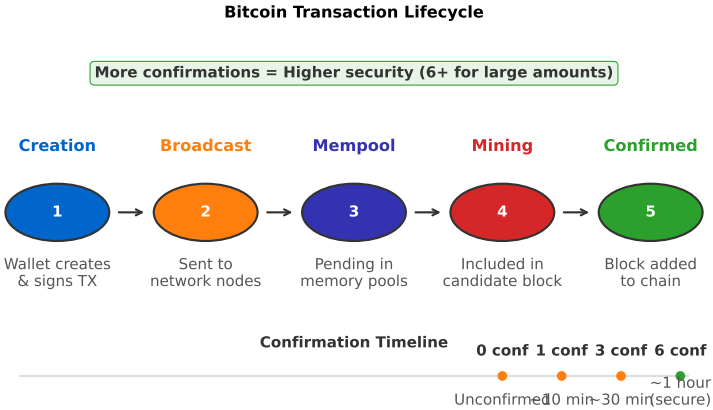
- Transaction lifecycle from creation to confirmation
- How do miners decide which transactions to include?

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 prevent digital..."?

What Happens to a Transaction from Creation to Confirmation?



Key Stages: Creation → Signing → Broadcast → Mempool → Mining → Confirmation

Each stage adds verification – unconfirmed transactions can still be double-spent

What Are the Key Stages in Transaction Processing?

1. Creation and Signing:

- Wallet selects UTXOs, constructs inputs/outputs, calculates fee
- Signs each input with corresponding private key

2. Broadcast and Mempool:

- Transaction sent to connected nodes, validated, added to mempool
- Nodes relay to peers (propagation across network)
- Transactions sorted by fee rate in mempool

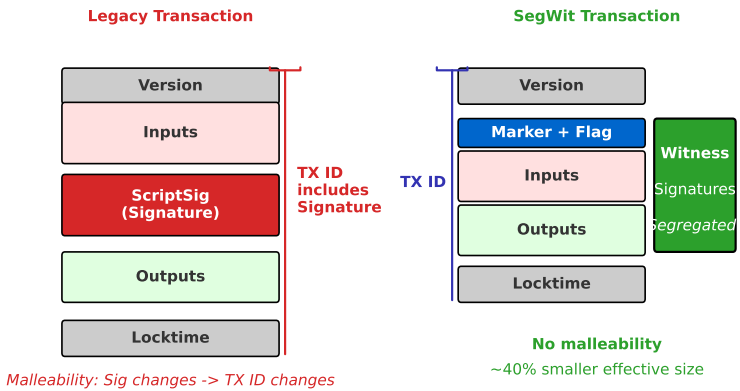
3. Mining and Confirmation:

- Miner includes transaction in candidate block
- Block mined, broadcast, validated by network
- Each new block adds one confirmation
- 6 confirmations typically considered final ($\sim$1 hour)

Key point: 1. Creation and Signing

How Did SegWit Fix Transaction Malleability?

Segregated Witness: Separating Signatures



Problem with Legacy: Signature in TX hash → malleability
SegWit Solution (BIP 141, 2017): Separate witness data from TX ID

What Benefits Did SegWit Bring to Bitcoin?

Block Capacity:

- Legacy: 1 MB limit
- SegWit: 4M weight units
- Witness: 1 byte = 1 WU
- Effective: 2-2.7 MB/block

Lower Fees:
Witness data 75% discount

Address Formats:

- P2WPKH: "bc1q..." (Bech32)
- P2SH-wrapped: "3..."

Enables Lightning:
Fixes malleability for payment channels

Compare the approaches shown above

What Improvements Did Taproot Introduce?

Key Improvements:

- **Schnorr:** Signature aggregation
- **MAST:** Hidden scripts until used
- **Privacy:** Uniform tx appearance

Address: "bc1p..." (Bech32m)

Benefits:

- Multi-sig = single-sig appearance
- Complex contracts look simple
- Smaller transaction size

Compare the approaches shown above

How Are Transactions Validated by Nodes?

Syntax:

- Size within limits
- Values non-negative
- No duplicate inputs

Semantic:

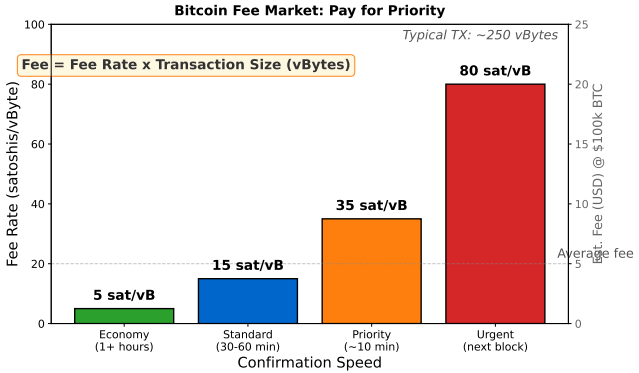
- UTXOs exist, unspent
- Valid signatures
- Scripts succeed
- Locktime satisfied

Rejection:

- Invalid sig = fraud
- Double-spend
- Dust output (spam)

→ Problem: How do we prevent digital... — Transaction Validation Rules — Validation rules are the network's defense against counterfeiting – invalid transactions rejected by all nodes

How Does Bitcoin's Fee Market Work?



Fee Market Dynamics:

- Block space is scarce (∼4 MB weight per 10 minutes)
- Users compete for inclusion via fees
- Fee estimation based on mempool state and target confirmation time

Key point: Fee Market Dynamics

How Can You Speed Up Stuck Transactions?

RBF (BIP 125):

- Replace tx with same inputs
- Increase fee (+1 sat/byte)
- Signal via sequence number
- Sender controls

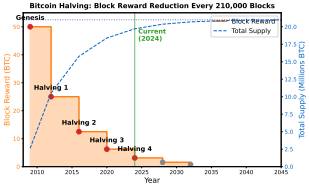
CPFP:

- Spend unconfirmed output
- High-fee child tx
- Miners include both
- Receiver controls

Key Difference: RBF = sender bumps; CPFP = receiver bumps

Compare the approaches shown above

How Do Miners Get Paid for Securing the Network?



Coinbase Properties:

- First transaction in every block, creates new bitcoins
- Miner collects block reward + all transaction fees
- Must wait 100 confirmations before spending (maturity rule)

Key point: Coinbase Properties

How Have Bitcoin ETFs Changed the Landscape?

Jan 10, 2024: ETF Approval

- SEC approved 11 spot ETFs
- BlackRock, Fidelity, Grayscale
- $140B+ AUM (Jan 2026)

Market Impact:

- Institutional legitimization
- Rivals major commodity ETFs
- Regulated custody

Transaction Implications:

- Large on-chain tx for ETFs
- Institutional UTXO consolidation
- Increased block space demand

Compare the approaches shown above

Problem Solved? Bitcoin's Transaction Protocol

The Original Problem

How do we prevent digital counterfeiting?

How Bitcoin's Protocol Solves It

- Tracks coin ownership cryptographically – each UTXO has a unique digital signature proving ownership, making counterfeiting computationally infeasible
- Create economic incentives for miners to include transactions in blocks, ensuring the system processes payments without centralized payment processors
- Bootstrap network security by rewarding early miners, gradually transitioning to fee-only economy by 2140

Remaining Limitations

- 7 transactions per second (TPS) limit vs Visa's 24,000 TPS – Bitcoin sacrificed scalability for decentralization and security
- 10-minute average block time means 1-hour wait for 6 confirmations, making point-of-sale payments impractical

Open Questions

- **Post-2140:** How will the network maintain security when all 21M coins are mined and only transaction fees remain? Will fees alone suffice?
- Sybil attacks, eclipse attacks, quantum computing threat to ECDSA signatures

Bitcoin's transaction protocol solves digital counterfeiting but introduces throughput and latency trade-offs

The Economics of Mining Economics

Incentive Structure

- Coordinating without trusted intermediaries
- Participants verify rather than trust
- Users gain trustlessness, pay in complexity

Economic Security

- Attack cost must exceed potential gain
- Honest behavior = Nash equilibrium

Key Economic Question

Who Pays, Who Earns?

Users gain trustlessness, pay in complexity

Design Principle

$$\text{Attack Cost} > \text{Potential Gain}$$

Cryptoeconomic security: Honest behavior must be the Nash equilibrium

Design Space: Mining Economics

Alternatives Considered

1. Permissioned vs permissionless
2. Traditional trusted third parties

Trade-offs Made

- Every design optimizes some properties
- ... at the expense of others

Design Questions

- What would YOU change?
- What's optimized? What's sacrificed?
- Are there other approaches?

Key Insight

No Perfect Solution

All blockchain designs involve trade-offs between decentralization, security, and scalability.

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

Failure Modes

Failure Modes: Mining Economics

Critical Failure Mode

- **What breaks:** Sybil attacks, eclipse attacks
- **Why it happens:** Economic incentives misaligned

Root Cause

- Assumption violated
- Incentive structure broken
- External shock

Historical Context

- Multiple real-world failures documented
- Patterns repeating across protocols

Early Warning Signs

- ! Unusual economic behavior
- ! Incentive misalignment
- ! Centralization drift

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

Script Puzzle Complexity

[COMIC: A Bitcoin Script as a Rube Goldberg machine – signatures push data onto stacks, operations verify and transform, finally a green "TRUE" light illuminates. A developer sighs: "All that just to prove I own 0.001 BTC." Caption: "Simple concept, complex execution."]

[Comic placeholder – to be illustrated]

- Bitcoin Script is intentionally limited – no loops, no Turing-completeness
- This simplicity is a security feature: predictable execution, no infinite loops

Key point: Script Puzzle Complexity

Key Takeaways

- UTXO model: consume old, create new
- Inputs + outputs structure
- Bitcoin Script: flexible, not Turing-complete
- P2PKH → P2SH → SegWit → Taproot
- Lifecycle: create → sign → broadcast → mine
- Fee market competition
- SegWit/Taproot: scalability + privacy

Design Philosophy: Security and decentralization over throughput

Next Lesson: L07 – Proof of Work

Compare the approaches shown above

Discussion Questions

1. Why does Bitcoin use the UTXO model instead of the account model?
2. How does the fee market incentivize miners to include transactions?
3. What are the trade-offs between legacy addresses, SegWit, and Taproot?
4. How does transaction malleability affect second-layer solutions?
5. Why is the coinbase maturity rule (100 confirmations) necessary?
6. How could you design a transaction that can only be spent after a certain date?

Key point: Discussion Questions

Continued

Next Lesson Preview: L07 Proof of Work Consensus

Topics to be covered:

- Mining mechanics and the proof-of-work algorithm
- Nonce searching and difficulty adjustment
- Block header structure and hash rate
- 51% attacks and mining centralization risks
- Energy consumption and environmental concerns

Preparation:

- Review hash function properties (pre-image resistance)
- Explore Bitcoin mining pools and hash rate distribution
- Consider the economics of mining profitability

Key point: Topics to be covered

Quiz - Part 1

Q1. What does UTXO stand for?
A) Unified Transaction Exchange Output B) Unspent Transaction Output C) Universal Token Exchange Order D) Unverified Transaction Object
Answer: B – UTXO = Unspent Transaction Output, representing spendable bitcoin.

Q2. How is a Bitcoin transaction fee calculated?
A) Fixed percentage of amount B) Sum of inputs minus sum of outputs C) Set by miners D) Based on recipient address
Answer: B – Fee = Total Input Value - Total Output Value (implicit, not explicitly stated).

Q3. What is the purpose of the ScriptSig in a transaction input?
A) Lock the output B) Set the transaction fee C) Provide proof of ownership (signature) D) Specify the recipient
Answer: C – ScriptSig provides the unlocking script with signature proving ownership of the UTXO.

Q4. How many satoshis are in one Bitcoin?
A) 1,000 B) 1,000,000 C) 100,000,000 D) 1,000,000,000
Answer: C – 1 BTC = 100,000,000 satoshis (eight decimal places).

Quiz - Part 2

Q5. Which statement about Bitcoin Script is correct?
A) It is Turing-complete B) It uses a stack-based execution model C) It supports loops D) It runs on Ethereum
Answer: B – Bitcoin Script is stack-based and intentionally NOT Turing-complete (no loops).

Q6. What does P2PKH stand for?
A) Pay to Protocol Key Hash B) Pay to Public Key Hash C) Private to Public Key Handler D) Peer to Peer Key Hash
Answer: B – P2PKH = Pay to Public Key Hash, the standard legacy transaction type.

Q7. What problem did SegWit solve?
A) Block size limit B) Transaction malleability C) Mining centralization D) Key generation
Answer: B – SegWit separates witness data from txid, fixing transaction malleability.

Q8. What prefix do native SegWit (Bech32) addresses start with?
A) 1 B) 3 C) bc1q D) 0x
Answer: C – Native SegWit addresses use Bech32 encoding starting with "bc1q".

Quiz - Part 3

Q9. How many confirmations are typically considered final for a Bitcoin transaction?
A) 1 B) 3 C) 6 D) 10
Answer: C – 6 confirmations ($\sim$1 hour) is the standard for high-value transaction finality.

Q10. What is RBF (Replace-by-Fee) used for?
A) Reduce transaction size B) Increase fee to speed up confirmation C) Cancel already confirmed transactions D) Create multi-sig wallets
Answer: B – RBF allows replacing an unconfirmed transaction with a higher-fee version.

Q11. What is a coinbase transaction?
A) Transaction on Coinbase exchange B) First transaction creating new BTC in each block C) Multi-signature transaction D) Fee refund transaction
Answer: B – Coinbase transaction is the first tx in each block, creating the block reward.

Q12. How long must a miner wait before spending coinbase outputs?
A) 10 blocks B) 50 blocks C) 100 blocks D) 144 blocks
Answer: C – Coinbase maturity rule requires 100 confirmations before spending.

Quiz

Quiz - Part 4

Q13. What did Taproot introduce to Bitcoin (activated 2021)?
A) Proof of Stake B) Schnorr signatures and MAST C) Smart contract execution D) Larger block size
Answer: B – Taproot introduced Schnorr signatures and Merkelized Abstract Syntax Trees (MAST).

Q14. In the UTXO model, what happens to your "balance" after spending?
A) It decreases incrementally B) Old UTXOs are consumed, new ones created C) Balance is updated in global state D) Nothing changes
Answer: B – UTXOs are consumed entirely; change creates new UTXOs.

Q15. Why is Bitcoin's UTXO model considered more privacy-friendly than account models?
A) UTXOs are encrypted B) No global balance visible, each UTXO is independent C) Addresses are hidden D) Transactions are anonymous
Answer: B – UTXO model has no visible balance; coins can be spread across many addresses.

Q16. What is a "dust" output in Bitcoin?
A) A high-value transaction B) An output too small to be economically spent C) A mining reward D) A test transaction
Answer: B – Dust outputs are so small that the fee to spend them exceeds their value.