
Propreties of
Ethereum

Table of contents

Motivation & Roadmap

Architecture

Account-based global state
Faster blocks under decentralization constraints

Execution

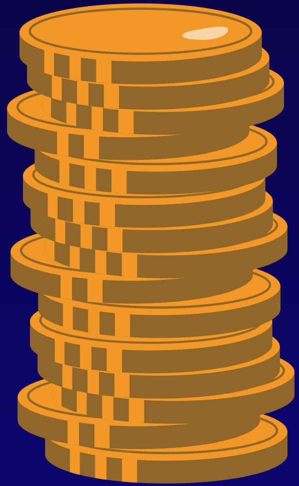
Turing-complete EVM

Persistent smart contract logic

Safety

Gas-metered execution

DoS and non-termination prevention



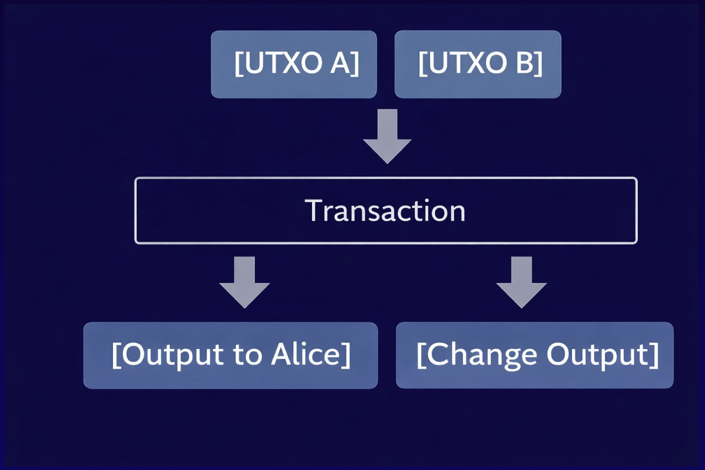
Bitcoin baseline : UTXO +constrained scriptes

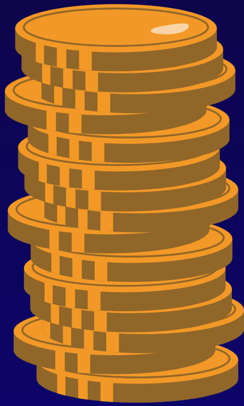
State Model(UTXO)
Digital cash abstraction

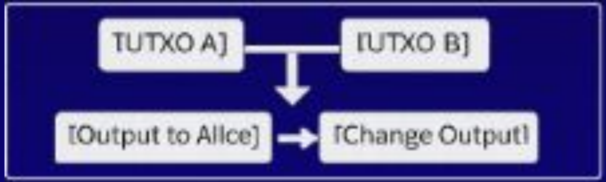
- State set of unspent transaction outputs
- Each output is spent entirely not at all

Trasaction Logic
Consume -retracte

- Inputs consume various UTXO's
- Output create:
    - Recipient value
    - Change output







Why this design works
Strengths of Bitcoin's baseline

Highly composable

Naturally parallelizable

Easy to validate securely at scale

Account-Based State

Account-Based Model

Ethereum replaces UTXOs with accounts

Global state = set of accounts, not unspent outputs

Account Structure

Balance (ETH)

Nonce (replay protection)

Storage (persistent key–value state)

Code (for contracts only)

Persistent State

Contracts can store data across transactions Enables:

multi-step agreements

escrow & auctions

on-chain state machines

Trade-off

Persistent state grows over time

Storage must be priced and managed (→ gas)

Why Accounts Matter: Stateful dApps

UTXO Friction
Outputs are only spent or unspent
No native way to store:
Contract progress
Intermediate decisions
Complex logic requires awkward workarounds

Account-State Clarity

Native persistence
Contracts maintain internal state
Can store:
stages of execution
commitments & balances
reputations or flags

Multi-Step Agreements

Where accounts shine

Escrow contracts

Auctions (bidding → reveal → settlement)
Hedging / financial contracts

Trade-off
Persistent state is not free

Step 1 → Update State → Final Settlement

Trade-off
important

Step 1 → Step 2 → Step 3
Deposit Update State Final Settlement

Improvement #2: Modified GHOST + Uncles

Problem with Bitcoin's PoW
Stale work is discarded
Stale blocks receive no reward
Valid proof-of-work is wasted
Faster blocks amplify this inefficiency
Creates centralization pressure

Ethereum's Design Choice
Include stale work
Inspired by GHOST (Greedy Heaviest Observed Subtree)
Blocks can reference uncles (stale blocks)
Stale blocks still contribute to chain weight

Why Uncles Matter
Security under fast block times
Reduces wasted hash power
Limits large-pool advantage
Keeps mining more decentralized
Security reflects actual work performed

Why Faster Blocks Create a Security Problem

Design Goal

Ethereum targets short block times

Goal: improve usability for smart contracts and dApps

Network Reality

Blocks need time to propagate across the network

Short block times ⇒ more competing blocks

Stale Blocks

Higher fork rate

Many valid blocks become stale

Work is wasted

Centralization Pressure

Unequal impact
Large miners / pools:
Faster propagation
Lower stale risk
Small miners:
Higher stale probability
More wasted work
⟶ Economic advantage concentrates hashing power

Incentives: Paying for Uncles

Why Incentives Matter

Latency creates unfair advantages

Network delays create stale blocks

Without rewards, stale work is pure loss

Large pools suffer less from latency than small miners

Ethereum’s Incentive Design

Uncle miners receive a significant fraction of a block reward

Including miners earn a small bonus for referencing uncles

Stale blocks become economically relevant

Economic Effect

Less wasted computation

Smaller miners stay competitive

Lower incentive to join only large pools

The Merge PoW → PoS

Ethereum as an Evolving Protocol

Ethereum has continuously evolved since launch

Design choices adapt to new constraints and goals

The Merge (September 2022)

Transition from Proof of Work (PoW) to Proof of Stake (PoS)

Consensus mechanism changed

Block production and security assumptions updated

What Changed and What Didn’t Changed:

Mining → staking

Energy consumption

Validator incentives

Unchanged (key point):

Account-based state

Smart contracts

EVM execution model

Gas as execution constraint

Turing-Complete EVM vs Bitcoin Script

🟠 Bitcoin Script

Constrained by design

Stack-based scripting language

No general loops

Limited control flow

No persistent contract state

→ Predictable and safe verification

⚠️ Theoretical Implication

The halting problem

Turing-complete programs may not terminate

No general algorithm can decide if a program halts

On a blockchain, non-termination is a security risk

🔵 Ethereum Virtual Machine (EVM)

General-purpose execution

General control flow (conditionals, iteration)

Persistent contract state

Contract-to-contract calls

Designed to be Turing-complete

→ Arbitrary computation is possible

Gas: Metering Execution

⛔ No Halting Solution

Ethereum does not solve the halting problem

Turing-complete execution allows non-terminating programs

Ethereum does not try to detect termination

Instead, it prices computation

⚙️ Metered Execution

Gas is charged per:

Opcode execution

Memory expansion

State access

⛽ Gas as a Computation Budget

Each transaction sets a gas limit

Gas = maximum amount of computation the sender is willing to pay for

Every EVM operation consumes gas

❌ Out-of-Gas (OOG) Behavior

Execution is immediately aborted

State changes are reverted

Gas already spent is not refunded

→ Infinite loops become economically impossible

EIP-1559: Gas Pricing

⚙️ What EIP-1559 Changes

Modifies how gas is prices

Improves fee predictability

Reduces fee volatility

🔥 Base Fee

Protocol-controlled

Adjusts automatically with network congestion

Is burned (removed from supply)

Makes fees more predictable for users

💰 Transaction Fee Structure

Fee = gasUsed × *effectiveGasPrice*

effectiveGasPrice is composed of:

Base fee (set by protocol, burned)

Priority fee (tip to validator)

⛽ What Does NOT Change

Key reminder

Gas limit still sets a hard execution bound

Gas remains the safety mechanism for

Turing-complete execution

Resources

- Bitcoin: A Peer-to-Peer Electronic Cash System (bitcoin.pdf)
- Bitcoin Developer Docs — Transactions
- Buterin — Ethereum: A Next-Generation Smart Contract and dApp Platform
- Wood — Ethereum Yellow Paper
- EIP-1559 — Fee market change