Lesson 19: Ethereum and Smart Contracts

Module 2: Blockchain Fundamentals

Digital Finance

Bitcoin's Limitations: Why Ethereum?

Bitcoin Script:

- Not Turing-complete (no loops)
- Limited expressiveness
- Designed for simple transfers
- No complex state

Ethereum Vision (Vitalik Buterin, 2013):

- Turing-complete programming
- Decentralized applications (dApps)
- "World Computer"
- Programmable money and agreements

charts/lesson_19/bitcoin_vs_ethereum.pdf

Smart Contracts: Code as Law

charts/lesson_19/smart_contract_concept.pdf

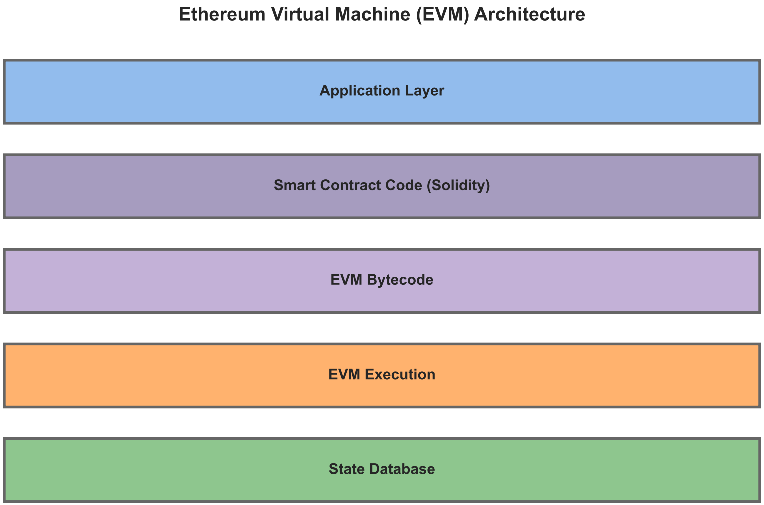
Account Model: Ethereum's Design

Two Account Types:

1. **Externally Owned Accounts (EOAs):**
   - Controlled by private key
   - Can send transactions
   - No code
2. **Contract Accounts:**
   - Controlled by code
   - Triggered by transactions
   - Store state

`charts/lesson_19/account_types.pdf`

Ethereum Virtual Machine (EVM)



Gas: Metering Computation

Why Gas?

- Prevent infinite loops (halting problem)
- Prioritize transactions
- Compensate miners/validators
- Align incentives

Gas Mechanics:

- Each operation costs gas
- User sets gas limit + gas price
- Unused gas refunded
- Out of gas → revert (but gas consumed)

charts/lesson_19/gas_mechanism.pdf

Gas Costs: Operation Examples

Operation	Gas Cost	Rationale
ADD (arithmetic)	3	Simple computation
MUL (multiplication)	5	Slightly more complex
SSTORE (write storage)	20,000	Permanent state change
SLOAD (read storage)	2,100	Storage access
CREATE (deploy contract)	32,000	Base cost + code size
Transaction (base)	21,000	Minimum for any transaction

Design: Expensive operations (storage, deployment) cost more to prevent spam

Transaction Cost Calculation

Example: Simple ETH Transfer

- Gas limit: 21,000
- Gas price: 50 gwei (1 gwei $= 10^{-9}$ ETH)
- Total fee: $21,000 \times 50 \times 10^{-9} = 0.00105$ ETH

Example: Token Transfer (ERC-20)

- Gas limit: 65,000 (contract interaction)
- Gas price: 50 gwei
- Total fee: $65,000 \times 50 \times 10^{-9} = 0.00325$ ETH

Example: Complex DeFi Swap

- Gas limit: 300,000 (multiple contract calls)
- Gas price: 100 gwei (priority)
- Total fee: $300,000 \times 100 \times 10^{-9} = 0.03$ ETH ($\sim$\$60 at \$2000/ETH)

Legacy Fee Market: First-Price Auction

Pre-EIP-1559 (before Aug 2021):

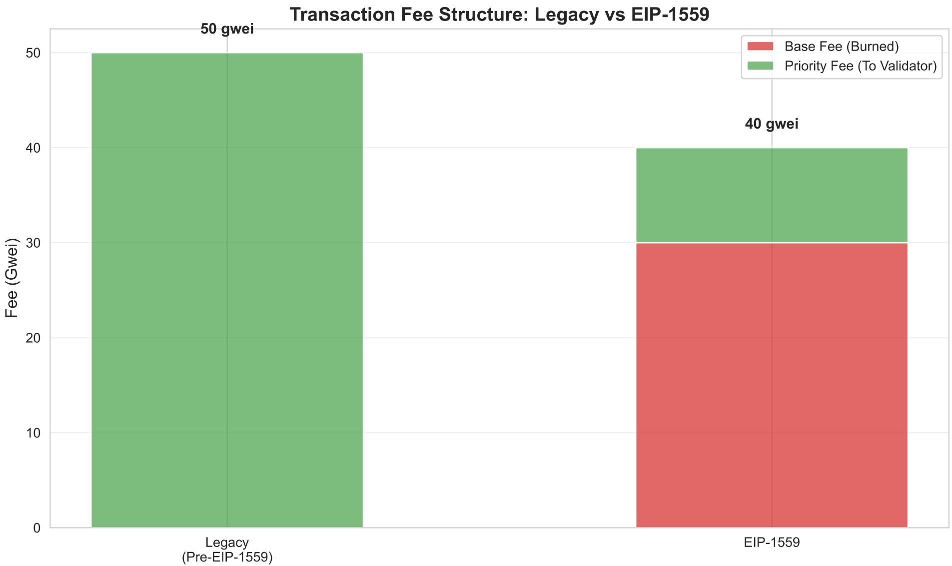
- Users bid gas price
- Miners select highest bids
- First-price auction
- Overpay or get stuck

Problems:

- Fee estimation difficult
- High volatility
- Miner extractable value (MEV)

charts/lesson_19/legacy_fee_market.pdf

EIP-1559: Fee Market Reform (August 2021)



New Fee Structure:

EIP-1559 Base Fee Dynamics

Base Fee Adjustment:

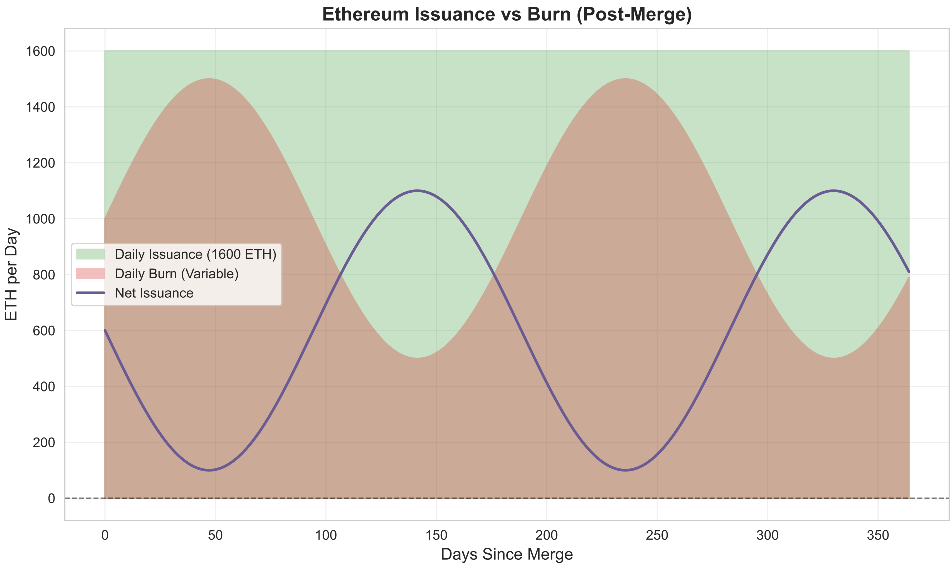
- Target: 15M gas per block
- If block $>$ 15M: base fee $\uparrow$ 12.5%
- If block $<$ 15M: base fee $\downarrow$ 12.5%
- Max block size: 30M gas

Formula:

$$\Delta_{\text{base}} = \frac{\text{Gas Used} - 15M}{15M} \times \frac{\text{Base Fee}}{8}$$

charts/lesson_19/base_fee_adjustment.pdf

Fee Burning: Deflationary Pressure



Issuance: ~1,600 ETH/day (PoS rewards)

Example: Simple Token Contract

charts/lesson_19/solidity_example.pdf

Smart Contract Lifecycle

charts/lesson_19/contract_lifecycle.pdf

Contract Storage: Persistent State

Storage Layout:

- Key-value store (256-bit slots)
- Permanent (persists between calls)
- Expensive (SSTORE = 20,000 gas)
- Optimizations: packing, mappings

Memory vs Storage:

- **Storage:** Permanent, expensive
- **Memory:** Temporary, cheap, cleared after execution

charts/lesson_19/storage_vs_memory.pdf

Events and Logs

Purpose:

- Emit structured data from contracts
- Stored in transaction receipts
- Indexed for efficient querying
- Off-chain applications listen to events

Use Cases:

- Token transfers (Transfer event)
- Price updates (oracles)
- Audit trails
- UI updates (wallets, dApps)

charts/lesson_19/events_architecture.pdf

External Calls: Composability and Risks

Contract Interactions:

- Contracts can call other contracts
- Enables composability ("money legos")
- DeFi protocols build on each other

Risks:

- Reentrancy attacks
- Uncontrolled gas consumption
- Malicious contract logic
- Dependency vulnerabilities

charts/lesson_19/external_call_flow.pdf

Reentrancy: The DAO Hack (2016)

charts/lesson_19/reentrancy_attack.pdf

Oracles: Bridging On-Chain and Off-Chain

Problem:

- Smart contracts cannot access external data
- No internet, APIs, randomness
- Determinism requirement

Oracle Solution:

- Third-party data feeds
- Price feeds (ETH/USD)
- Weather data
- Sports scores

charts/lesson_19/oracle_architecture.pdf

Chainlink: Decentralized oracle network, aggregates multiple data sources

Gas Optimization Strategies

- **Storage Packing:** Use uint128 instead of uint256 where possible (fit in one slot)
- **Avoid Storage Writes:** Use memory for temporary data
- **Short-Circuit Logic:** `require` checks early, minimize wasted gas
- **Batch Operations:** Aggregate multiple actions in one transaction
- **Events over Storage:** Emit events instead of storing historical data
- **Minimal Contract Size:** Lower deployment costs
- **Use Libraries:** Reusable code via DELEGATECALL

Example: Storing 100 values individually: $\sim$2M gas. Packed into single array: $\sim$500K gas

Upgradeable Contracts: Proxy Pattern

charts/lesson_19/proxy_pattern.pdf

Summary

- **Ethereum:** World computer, Turing-complete smart contracts, account model
- **EVM:** Stack-based VM, executes bytecode, deterministic, replicated
- **Gas:** Meters computation, prevents infinite loops, aligns incentives
- **EIP-1559:** Base fee (burned) + priority fee, deflationary pressure
- **Solidity:** High-level language, compiles to bytecode
- **Risks:** Reentrancy, oracles, immutability challenges

Next Lesson: Tokens – ERC-20, ERC-721 (NFTs), and token economics