# Lesson 07 — Smart Contracts & Game Theory

Study Notes

Cryptoeconomics — BSc Level

Joerg Osterrieder

2025

## Contents

## Learning Objectives

By the end of this lesson, students should be able to:

1. **Define** smart contracts and explain the EVM execution model, gas, and the checks-effects-interactions security pattern.
2. **Represent** a game in normal form and identify dominant strategies, Nash equilibria, and mixed-strategy equilibria.
3. **Compute** Nash equilibria for $2 \times 2$ payoff matrices and interpret them in the context of blockchain protocols.
4. **Explain** how the Prisoner's Dilemma applies to miner and validator coordination problems.
5. **Define** mechanism design and incentive compatibility; give examples from PoW, PoS, and oracle systems.
6. **Describe** quadratic voting, its cost structure, and its application in Gitcoin Grants.
7. **Analyze** the veToken model (vote-escrow, time decay, gauge voting) and explain the Curve Wars.
8. **Characterize** MEV as a game-theoretic problem and evaluate mitigation strategies including PBS and Flashbots.
9. **Compare** auction types and explain EIP-1559 as an auction mechanism design improvement.
10. **Identify** the reentrancy vulnerability, integer overflow, and access control failures in Solidity, and describe audit best practices.

## Smart Contracts

### Definition and Properties

**Smart Contract**

A **smart contract** is a program stored on a blockchain that executes automatically when predefined conditions are met, without requiring an intermediary. The code is deployed once and runs deterministically on every node that processes the containing transaction.

Key properties:

- **Deterministic**: Given the same inputs and state, every node produces identical outputs.
- **Immutable**: Once deployed, the bytecode at an address cannot be changed (proxy patterns simulate upgrades by delegating to a swappable logic contract).
- **Transparent**: Source code (when verified) and state are publicly readable.
- **Trustless**: Executes exactly as written; no intermediary can alter the outcome.

## Solidity and EVM Execution Model

Solidity is the primary language for Ethereum smart contracts. It compiles to **EVM bytecode**, which runs on the **Ethereum Virtual Machine**.

**EVM key concepts:**

- **Stack-based VM**: Operations push and pop 256-bit words on a last-in-first-out stack (max depth 1024).
- **Storage**: Persistent key-value store (expensive: 20,000 gas for a new slot). State variables live here.
- **Memory**: Transient, cleared after each call (cheaper).
- **Calldata**: Read-only input data for external calls.
- **Gas**: Each opcode costs a fixed amount of gas. The transaction specifies a gas limit; execution reverts if it is exceeded.

**Minimal Solidity example:**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract SimpleVault {
    mapping(address => uint256) public balances;

    function deposit() external payable {
        balances[msg.sender] += msg.value;      // Effects first
    }

    function withdraw(uint256 amount) external {
        require(balances[msg.sender] >= amount, "Insufficient");
        balances[msg.sender] -= amount;         // Effect BEFORE interaction
        (bool ok,) = msg.sender.call{value: amount}("");
        require(ok, "Transfer failed");
    }
}
```

### Token Standards

| Standard | Type | Use Case |
|---|---|---|
| ERC-20 | Fungible tokens | USDC, DAI, UNI, AAVE; requires `transfer`, `approve`, `transferFrom` |
| ERC-721 | Non-fungible tokens | Digital art, game items; each token has a unique ID |
| ERC-1155 | Multi-token | Games with mixed fungible/non-fungible assets; batch transfers |
| ERC-4626 | Tokenized vaults | Standard interface for yield-bearing tokens (e.g., Aave aTokens) |

## Game Theory Foundations

**Game Theory**

**Game theory** is the mathematical study of strategic interaction among rational agents. A **game** in normal form is defined by: (1) a set of *players* $N = \{1, \ldots, n\}$; (2) for each player $i$ a set of *strategies* $S_i$; (3) for each strategy profile $(s_1, \ldots, s_n) \in S_1 \times \cdots \times S_n$ a *payoff* $u_i(s_1, \ldots, s_n)$ for player $i$.

### Dominant Strategies

Strategy $s_i$ **strictly dominates** $s_i'$ if for all strategy profiles of opponents:

$$u_i(s_i, s_{-i}) > u_i(s_i', s_{-i}) \quad \forall s_{-i}$$

A rational player always plays a strictly dominant strategy if one exists. **Iterated elimination of dominated strategies (IEDS)**: repeatedly remove dominated strategies until no more remain. The surviving strategies are *rationalizable*.

### Mixed Strategies

A **mixed strategy** $\sigma_i$ is a probability distribution over $S_i$. Player $i$ maximizes expected payoff:

$$U_i(\sigma_i, \sigma_{-i}) = \sum_{s \in S} \sigma_1(s_1) \cdots \sigma_n(s_n) \cdot u_i(s)$$

Mixed strategies model randomized behavior when no pure dominant strategy exists and are essential for games like penalty kicks or protocol randomness.

### Nash Equilibrium

**Nash Equilibrium (Nash 1950)**

A strategy profile $(s_1^*, \ldots, s_n^*)$ is a **Nash equilibrium** if no player can unilaterally increase their payoff by deviating:

$$\forall i,\ \forall s_i \in S_i : \quad u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$$

Every finite game has at least one Nash equilibrium (possibly in mixed strategies).

## Computing Nash Equilibria for $2 \times 2$ Games

Given the payoff matrix (row player payoffs first, column player second):

| | | **Column Player** | |
|---|---|---|---|
| | | **L** | **R** |
| **Row Player** | **U** | $(a, b)$ | $(c, d)$ |
| | **D** | $(e, f)$ | $(g, h)$ |

**Step-by-step procedure:**

1. Find Row's best responses: compare $(a, e)$ for Column playing L; compare $(c, g)$ for Column playing R. Mark dominant entries.
2. Find Column's best responses: compare $(b, d)$ for Row playing U; compare $(f, h)$ for Row playing D.
3. A cell $(i, j)$ is a pure Nash equilibrium if *both* players are best-responding.
4. If no pure equilibrium exists, solve for mixed equilibrium by making each player indifferent between their strategies.

## Multiple Equilibria

Many games have multiple Nash equilibria. This creates a **coordination problem**: which equilibrium will players select? **Schelling focal points** — naturally salient equilibria — help coordinate expectations.

**Example: Coordination in Proof-of-Stake**

Two validators simultaneously decide whether to attest to block $B$ or block $B'$ (a competing fork). The payoff matrix (symmetric):

| | **Attest** $B$ | **Attest** $B'$ |
|---|---|---|
| **Attest** $B$ | $(10, 10)$ | $(-5, -5)$ |
| **Attest** $B'$ | $(-5, -5)$ | $(10, 10)$ |

Both $(B, B)$ and $(B', B')$ are Nash equilibria. The canonical chain is the Schelling focal point; slashing for equivocation (attesting both) enforces it.

## Prisoner's Dilemma in Crypto

### The Classic Game

| | | **Player B** | |
|---|---|---|---|
| | | **Cooperate (C)** | **Defect (D)** |
| **Player A** | **Cooperate (C)** | $(-1, -1)$ | $(-3, 0)$ |
| | **Defect (D)** | $(0, -3)$ | $(-2, -2)$ |

**Analysis:**

- *Defect* strictly dominates *Cooperate* for each player.
- Unique Nash equilibrium: $(D, D)$ with payoff $(-2, -2)$.
- Pareto-optimal outcome: $(C, C)$ with payoff $(-1, -1)$.
- Individual rationality destroys collective welfare: the "dilemma."

### Miner Cooperation

In Bitcoin, miners could theoretically collude to fork the chain and double-spend (cooperative attack). Why don't they?

- A 51% attack requires enormous hashrate (capital cost).
- A successful attack destroys Bitcoin's value, making the BTC reward worthless.
- The dominant strategy is to mine honestly: block rewards are positive and certain; attack profits are uncertain and self-defeating.

The protocol transforms the Prisoner's Dilemma by making defection (attacking) strictly unprofitable, so cooperation (honest mining) becomes dominant.

### Staking Coordination

In Proof of Stake, validators deciding whether to stake face a staking-level coordination game:

- If too few stake: network is insecure; potential stakers should stake more to protect the value of their holdings.
- If too many stake: individual returns fall below opportunity cost; marginal validators withdraw.
- Equilibrium: staking rate where the marginal validator is indifferent between staking and holding.

The protocol calibrates staking rewards to achieve a target stake ratio (e.g., Ethereum targets 25–33% of circulating supply staked).

## Mechanism Design

### Mechanism Design

**Mechanism design** is "reverse game theory": given a desired social outcome, design rules and incentives such that self-interested rational agents playing the resulting game achieve that outcome. Also called the *engineering* branch of game theory.

### Incentive Compatibility

A mechanism is **incentive compatible (IC)** if truth-telling (or the desired action) is a dominant strategy for every participant:

$$u_i(\text{honest behavior}, s_{-i}) \geq u_i(s_i', s_{-i}) \quad \forall s_i', s_{-i}$$

**Examples:**

- **Proof of Work**: Mining a valid block earns the block reward. Mining an invalid block yields nothing (network rejects it). Honest mining is dominant.
- **Proof of Stake with slashing**: Attesting correctly earns rewards. Double-signing (attesting two conflicting blocks) triggers slashing (a large fraction of stake is burned). Honesty is dominant.
- **Chainlink oracles**: Node operators stake LINK. Reporting outliers leads to stake slashing. Reporting the consensus value (truth) maximizes expected return.

### Revelation Principle

The **Revelation Principle** states that for any mechanism, there exists an equivalent *direct mechanism* in which each agent truthfully reports their private information. This simplifies mechanism design: we can restrict attention to truthful, direct mechanisms without loss of generality.

### VCG Mechanism

The **Vickrey–Clarke–Groves (VCG)** mechanism is the canonical IC mechanism for social welfare maximization:

1. Each agent $i$ reports valuation $v_i$ for each outcome.
2. The mechanism selects the outcome maximizing $\sum_i v_i$.
3. Agent $i$ pays: $\sum_{j \neq i} v_j(\text{outcome without } i) - \sum_{j \neq i} v_j(\text{chosen outcome})$.

VCG is IC and achieves allocative efficiency. Drawbacks: revenue-suboptimal; computationally expensive at scale; collusion-prone.

## Quadratic Voting

### Cost Structure

**Quadratic Voting (Lalley & Weyl 2018)**

In **quadratic voting (QV)**, the cost of casting $v$ votes on an issue is $v^2$ tokens (or "voice credits"):

$$c(v) = v^2$$

The marginal cost of the $k$-th vote is $2k - 1$ tokens, increasing linearly with $k$. A voter with strong preferences buys many votes but pays a quadratically higher price.

| **Votes** $v$ | **Cost** $c(v) = v^2$ |
|---:|---:|
| 1 | 1 |
| 2 | 4 |
| 3 | 9 |
| 5 | 25 |
| 10 | 100 |

### Fairness Properties

Compared to one-token-one-vote (plutocracy) and one-person-one-vote:

- **Intensity representation**: QV allows voters to express the *strength* of preferences, not just direction.
- **Reduced whale dominance**: A voter with 100× more tokens has only 10× more votes, not 100×.
- **Incentive to reveal true intensity**: Under quadratic costs, a voter's dominant strategy is to buy votes proportional to their true valuation.

### Gitcoin Grants and Quadratic Funding

**Quadratic Funding (QF)** extends QV to public goods funding: the matching allocation for project $j$ is proportional to $\left(\sum_i \sqrt{c_{ij}}\right)^2$, where $c_{ij}$ is contributor $i$'s donation to project $j$.

- Many small donors produce a large match; single large donors produce a small match.
- Gitcoin Grants has distributed $67M+ to open-source public goods using QF since 2019.

**Common Pitfall**

**Sybil vulnerability**: QV and QF assume one identity per participant. A whale can split holdings across many wallets to gain disproportionate influence. Effective QV requires identity

verification (e.g., Passport, Proof of Humanity) or Sybil-resistance mechanisms.

## veToken Economics

### Vote-Escrow Model

**veToken (Vote-Escrow Token)**

In the **veToken model**, users lock governance tokens for a fixed period (up to 4 years) and receive non-transferable *vote-escrow* tokens (e.g., veCRV from CRV). Voting power and reward boost are proportional to the amount locked and the remaining lock duration.

Voting power for a lock of $L$ tokens for $t$ weeks (max $T = 208$ weeks):

$$\text{vePower}(L, t) = L \times \frac{t}{T}$$

As time passes, vePower decays linearly to zero at lock expiry. Users must re-lock to maintain voting power.

### Gauge Voting

Curve Finance uses veCRV to direct CRV emissions (liquidity incentives):

- Each liquidity pool has a **gauge** that receives a fraction of new CRV emissions.
- veCRV holders vote weekly on gauge weights, directing more rewards to pools they favor.
- Protocols compete to attract veCRV votes to their pools to increase their liquidity depth.

### The Curve Wars

The **Curve Wars** are the meta-game of protocols competing for CRV emissions:

1. Convex Finance accumulated massive veCRV by offering boosted yields to CRV depositors, becoming the largest veCRV holder.
2. Protocols (Frax, MIM, Alchemix) pay veCRV holders "bribes" via platforms like Votium or Hidden Hand to vote for their gauges.
3. The equilibrium: protocols effectively pay for liquidity through governance, not direct incentives. This is the "yield-as-governance" paradigm.

The Curve Wars demonstrate that governance systems can be captured through economic means even without a security exploit.

## MEV as Game Theory

### Definition and Sources

**Maximal Extractable Value (MEV)**

**MEV** is the maximum value that can be extracted by a block producer (miner or validator) beyond standard block rewards and fees, by reordering, including, or excluding transactions within a block.

**MEV types (by volume):**

- **DEX arbitrage**: Buy on a lower-priced DEX, sell on a higher-priced one within one block. Profitable when oracle or pool prices diverge.
- **Liquidations**: Race to liquidate undercollateralized positions first and capture the liquidation bonus.
- **Sandwich attacks**: Front-run a large pending DEX trade (buy before it, sell after), extracting slippage from the victim.
- **NFT sniping**: Front-run profitable NFT mints or purchases.

### Proposer-Builder Separation (PBS)

To reduce validator centralization from MEV, Ethereum separates roles:

- **Block builders** (sophisticated searchers and MEV extractors) assemble blocks maximizing MEV and bid for inclusion.
- **Block proposers** (validators) blindly select the highest-paying block without knowing its contents.

This prevents validators from needing sophisticated MEV infrastructure, reducing centralization pressure.

### Flashbots and Order Flow Auctions

**Flashbots** created MEV-Boost: a marketplace where builders bid for block production rights. Validators earn the winning bid; builders earn residual MEV. This:

- Reduces failed gas-war transactions in the public mempool.
- Redistributes some MEV to validators (and by extension stakers).
- Enables private transaction relays that protect users from front-running.

**Order Flow Auctions (OFAs)** let users auction their transaction order flow directly to searchers, capturing the MEV value for themselves rather than ceding it to block builders.

## Auction Theory

### Auction Types

| Type | Mechanism | Crypto Use Case |
|---|---|---|
| English (ascending) | Bidders raise price until one remains. Winner pays final bid. | NFT auctions (OpenSea), liquidation auctions. |
| Dutch (descending) | Price starts high and falls until a bidder accepts. | Token sales (e.g., Gnosis auction), rare NFT drops. |
| First-price sealed | Submit secret bids; highest bid wins at their bid. | Pre-EIP-1559 gas market. |
| Second-price sealed (Vickrey) | Submit secret bids; highest bid wins but pays second-highest price. | Theoretical base for incentive-compatible auctions. |

### Revenue Equivalence Theorem

Under standard assumptions (risk-neutral bidders, symmetric, independent private values), all four auction formats yield the same expected revenue to the seller. This is the **Revenue Equivalence Theorem** (Myerson 1981).

Deviations occur when:

- Bidders are risk-averse (favors first-price auctions).
- Values are correlated (e.g., common value auctions; favors second-price auctions).

### Winner's Curse

In **common value auctions** (each bidder has a noisy estimate of a common true value), the winning bidder tends to be the one who most overestimated the value — the *winner's curse*. Rational bidders shade their bids downward to compensate.

### EIP-1559 as Auction Mechanism Design

Pre-EIP-1559 Ethereum used a first-price sealed auction for gas. Flaws:

- Users overpaid (no refund for bidding above the inclusion threshold).
- Fee prediction was hard; users either under-bid (stuck) or over-bid (wasted money).

EIP-1559 replaced this with a **posted-price mechanism**:

- Base fee is algorithmically determined to target 50% block utilization. It rises 12.5% if the previous block was full, falls 12.5% if empty.
- Users set a `maxFeePerGas`. Inclusion is guaranteed iff `maxFeePerGas` $\geq$ base fee.
- Base fee is burned; validator receives only the priority tip.

This design is not fully incentive-compatible (validators can still manipulate by mining empty blocks to depress the base fee), but is robust for non-collusive validators.

## Smart Contract Security

### Reentrancy

**Reentrancy Vulnerability**

A **reentrancy attack** occurs when a contract sends ETH to an external address before updating its internal state. The receiving contract's fallback function calls back into the victim contract, which still shows the pre-update state, allowing repeated withdrawals.

**Classic example: The DAO Hack (2016)**

- $60M in ETH stolen via reentrancy.
- Led to a hard fork splitting Ethereum (ETH) from Ethereum Classic (ETC).

**Prevention: Checks-Effects-Interactions (CEI) pattern:**

1. **Checks**: Validate all preconditions (`require` statements).
2. **Effects**: Update all state variables.
3. **Interactions**: Call external contracts or send ETH.

Alternative: use a **reentrancy guard** (mutex) that reverts if the function is called while already executing (OpenZeppelin's `ReentrancyGuard`).

### Integer Overflow and Underflow

Prior to Solidity 0.8.0, arithmetic was unchecked. An **overflow** occurs when a value exceeds its maximum representation (e.g., `uint8` wraps $255 + 1 = 0$).

**Mitigation**:

- Solidity $\geq$ 0.8.0: arithmetic reverts on overflow/underflow by default.
- For older contracts: use OpenZeppelin's `SafeMath` library.
- Use `unchecked{}` blocks only when overflow is intended (e.g., loop counters).

### Access Control Failures

**Examples:**

- Missing `onlyOwner` or `onlyAdmin` modifiers on privileged functions (`mint`, `withdraw`, `upgrade`).
- Incorrect role initialization (admin role set to `address(0)`).
- Unprotected `selfdestruct` or `delegatecall` to user-supplied addresses.

**Best practices:**

- Use OpenZeppelin's `AccessControl` or `Ownable`.
- Use timelocks and multi-signature wallets for privileged actions.
- Apply the principle of least privilege: grant only necessary permissions.

### Audit Best Practices

1. **Internal review**: Run automated tools (Slither, Mythril, Foundry's `forge fuzz`) before external audit.
2. **External audit**: Commission at least two independent audit firms for production contracts holding significant value.
3. **Formal verification**: Use Certora Prover or SMTChecker for critical invariants (e.g., "total supply equals sum of balances").
4. **Bug bounty**: Deploy on Immunefi or HackerOne; set bounty proportional to TVL.
5. **Time-locked upgrades**: Use a governor + timelock so the community can review and cancel malicious upgrades.

**Common Pitfall**

**Audits do not guarantee security.** An audit is a point-in-time review. Contracts are still exploited after audits because: (1) auditors miss complex interactions with external protocols; (2) contracts are upgraded after the audit; (3) the threat model was incomplete. Always use bug bounties *and* post-deployment monitoring.

## Practice Problems

1. **(Nash Equilibrium)** Find all pure Nash equilibria of the following game. Show your work by underlining best responses.

| | | **Column** | |
|---|---|---|---|
| | | **Left (L)** | **Right (R)** |
| **Row** | **Up (U)** | $(3, 2)$ | $(1, 4)$ |
| | **Down (D)** | $(2, 1)$ | $(4, 3)$ |

*Solution:*
Row's best responses: if Column plays L, Row prefers U ($3 > 2$); if Column plays R, Row prefers D ($4 > 1$).
Column's best responses: if Row plays U, Column prefers R ($4 > 2$); if Row plays D, Column prefers R ($3 > 1$).
NE: $(D, R)$ with payoffs $(4, 3)$. Check: Row gets 4 (best vs. Column R: $4 > 1$). Column gets 3 (best vs. Row D: $3 > 1$). Confirmed.

2. **(Prisoner's Dilemma in Crypto)** Two mining pools $A$ and $B$ independently decide whether to include a fee-boosting cartel transaction. The payoff matrix (in BTC profit per block):

| | | **Pool B** | |
|---|---|---|---|
| | | **Include** | **Exclude** |
| **Pool A** | **Include** | $(5, 5)$ | $(8, 2)$ |
| | **Exclude** | $(2, 8)$ | $(6, 6)$ |

   a. Identify any dominant strategies.

   b. Find the Nash equilibrium.

   c. Is the Nash equilibrium Pareto optimal? Explain.

   *Solution:*

   a. For Pool A: Include dominates Exclude (8 ¿ 6 and 5 ¿ 2 in both columns). By symmetry, Include dominates for Pool B.

   b. Nash equilibrium: (Include, Include) with payoffs $(5, 5)$.

   c. No. $(Exclude, Exclude)$ yields $(6, 6)$, which Pareto-dominates $(5, 5)$. The pools could coordinate for mutual benefit but defection is individually rational.

3. **(Quadratic Voting)** A DAO uses QV with 100 voice credits per member. Alice values proposal X highly and wants to cast as many votes as possible while spending no more than 64 credits.

   a. How many votes can Alice cast? Show the cost calculation.

   b. Bob has 400 voice credits (accumulated over multiple rounds). How many votes can Bob cast on proposal X using at most 400 credits?

   c. Compare their voting influence: is QV more egalitarian than token-weighted voting?

   *Solution:*

   a. $v^2 \leq 64 \Rightarrow v \leq 8$. Alice casts 8 votes at cost $8^2 = 64$ credits.

   b. $v^2 \leq 400 \Rightarrow v \leq 20$. Bob casts 20 votes at cost 400 credits.

   c. Bob has $4\times$ more credits but only $20/8 = 2.5\times$ more votes. Under token-weighted voting, Bob would have $4\times$ more votes. QV is more egalitarian because the marginal cost of votes rises quadratically.

4. **(Mechanism Design / EIP-1559)** The base fee is currently 30 Gwei. The last block was 100% full (target: 50%).

   a. What is the new base fee in the next block?

b. A user sets `maxFeePerGas` = 35 Gwei and a priority tip of 2 Gwei. Their transaction uses 150,000 gas. How much ETH is burned, and how much goes to the validator?

c. If the network is congested for 10 consecutive 100%-full blocks starting from 30 Gwei, what is the approximate base fee after 10 blocks?

*Solution:*

a. Increase: $30 \times 1.125 = 33.75$ Gwei.

b. Base fee after adjustment: 33.75 Gwei < maxFeePerGas, so the transaction is included. Burned: $150{,}000 \times 33.75 \times 10^{-9} \approx 0.005063$ ETH. Validator: $150{,}000 \times 2 \times 10^{-9} = 0.0003$ ETH. (User is refunded: $(35 - 33.75 - 2) \times 150{,}000$ gas would be negative—tip absorbs remainder, so refund is from maxFee - actualBaseFee - priority = 35 - 33.75 - 2 = -0.75: tip is capped at maxFee - baseFee = 1.25 Gwei, not 2.) Correct: Validator receives $\min(2, 35 - 33.75) = 1.25$ Gwei $\times 150{,}000 = 0.0001875$ ETH.

c. After 10 blocks: $30 \times 1.125^{10} \approx 30 \times 3.247 \approx 97.4$ Gwei.

## Key Takeaways

1. **Smart contracts** are deterministic, immutable, transparent, and trustless. The EVM executes bytecode compiled from Solidity; gas meters computational effort. The CEI pattern and reentrancy guards are essential security primitives.

2. **Game theory** in normal form captures strategic interaction via players, strategy sets, and payoff functions. Dominant strategies and iterated elimination are the simplest solution concepts.

3. **Nash equilibrium** is a self-enforcing strategy profile: no player can profitably deviate unilaterally. Every finite game has at least one (possibly mixed). Multiple equilibria create coordination problems solved by focal points or protocol rules.

4. **The Prisoner's Dilemma** shows that individual rationality leads to collective inefficiency. Crypto protocols escape this by making defection unprofitable via slashing, opportunity cost, and protocol design.

5. **Mechanism design** engineers the game so the desired outcome is the Nash equilibrium. Incentive compatibility means truth-telling (or honesty) is dominant. The Revelation Principle simplifies design.

6. **Quadratic voting** balances intensity representation and whale resistance via $c(v) = v^2$. Sybil-resistance is required. QF extends QV to public goods funding.

7. **veTokens** align long-term incentives by time-locking governance power. The Curve Wars demonstrate that governance systems can be economically captured, creating the "yield-as-governance" paradigm.

8. **MEV** is a game where searchers, builders, and validators compete to extract value from transaction ordering. PBS and Flashbots democratize MEV extraction; OFAs return value to users.

9. **Auction theory** explains gas markets. EIP-1559 replaced inefficient first-price auctions with an algorithmic base fee plus priority tip, improving UX and creating deflationary pressure on ETH.

10. **Smart contract security** requires defense-in-depth: CEI patterns, automated analysis, external audits, formal verification, and bug bounties. Audits are necessary but not sufficient.

## Further Reading


- Buterin, V. (2014). *Ethereum Whitepaper*. https://ethereum.org/en/whitepaper/
  Original proposal for the EVM and smart contracts as a programmable blockchain.

- Solidity Documentation. https://docs.soliditylang.org/
  Official reference for the Solidity language, EVM opcodes, and security considerations.

- Nash, J. F. (1950). Equilibrium points in $n$-person games. *Proceedings of the National Academy of Sciences*, 36(1), 48–49.
  The two-page paper introducing Nash equilibrium.

- Roughgarden, T. (2021). *Transaction Fee Mechanism Design*. ACM SIGecom Exchanges, 19(1), 52–55. https://arxiv.org/abs/2106.01340
  Rigorous game-theoretic analysis of EIP-1559 as an auction mechanism.

- Lalley, S. P. & Weyl, E. G. (2018). Quadratic voting: How mechanism design can radicalize democracy. *AEA Papers and Proceedings*, 108, 33–37.
  The foundational paper on quadratic voting and its democratic properties.

- Daian, P. et al. (2020). Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. *IEEE S&P*. https://arxiv.org/abs/1904.05234
  Seminal paper introducing MEV and its implications for blockchain security.

- Consensys Diligence. *Smart Contract Best Practices*. https://consensys.github.io/smart-contract-best-practices/
  Comprehensive guide to Solidity security patterns.

- Mas-Colell, A., Whinston, M. D., & Green, J. R. (1995). *Microeconomic Theory*. Oxford University Press. Chapters 8–9 cover game theory; Chapters 23 cover mechanism design. The standard graduate reference.