Smart Contracts and Game Theory

Lesson 7: Designing Incentive-Compatible Systems

Prof. Joerg Osterrieder

Spring 2026

Learning Objectives

After this lesson, you will be able to:

- Explain why smart contracts need game theory to work safely
- Analyze blockchain interactions as strategic games
- Evaluate how mechanism design aligns selfish incentives with collective good
- Apply governance mechanism frameworks to real protocol design problems

Prerequisites: Lessons 1-6

Game theory explains why crypto protocols work

Lesson Outline

1. The Machine
2. The Game
3. The Design

The Machine

What is a Smart Contract?

Definition: Self-executing code that runs on a blockchain when conditions are met.

Key Properties:

- **Deterministic**: Same inputs produce same outputs
- **Immutable**: Code cannot be changed after deployment (though proxy patterns enable upgrades)
- **Transparent**: Code and state publicly visible
- **Trustless**: Executes without intermediaries

"Code is law" – smart contracts execute exactly as written

Smart Contract Capabilities

What Smart Contracts Can Do:

- Hold and transfer tokens
- Enforce rules and conditions
- Interact with other contracts
- Store data on-chain

What They Cannot Do:

- Access external data directly (need oracles)
- Execute without being triggered
- Modify themselves after deployment
- Reverse transactions once confirmed

Smart contracts are powerful but have limitations

The Oracle Problem

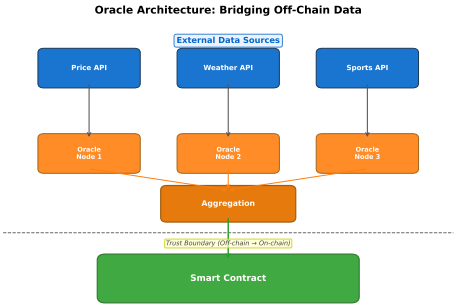
Challenge: Smart contracts cannot access off-chain data.

Examples Needing External Data:

- Price feeds (ETH/USD for liquidations)
- Sports results (betting contracts)
- Weather data (insurance contracts)
- Random numbers (lottery, gaming)

Solutions:

- Decentralized oracles (Chainlink)
- Multi-source aggregation
- Economic incentives for honesty

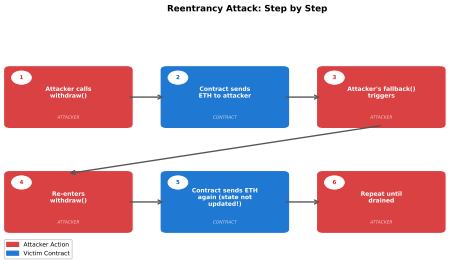


Oracles bridge the gap between blockchains and real world

Smart Contract Security: Reentrancy Attack

What is Reentrancy?

- Attacker's contract calls victim contract
- Victim sends ETH before updating state
- Attacker's fallback function calls victim again
- Victim sends ETH again (state not yet updated)
- Repeats until victim's funds are drained



Classic Example: The **DAO** hack (2016) – $60M stolen, led to Ethereum hard fork **Prevention: Checks-Effects-Interactions Pattern**

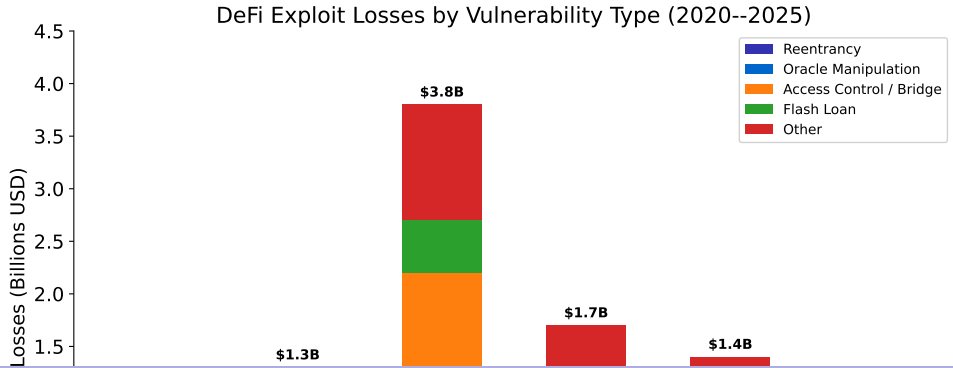
1. Checks: Validate conditions
2. Effects: Update state variables
3. Interactions: Call external contracts (after state update)

DAO hack analysis: hackingdistributed.com — **Security:** consensys.github.io/smart-contract-best-practices

Smart Contract Vulnerabilities Overview

Top Vulnerability Types and Notable Losses:

Vulnerability	Mechanism	Notable Losses
Reentrancy	Recursive callback before state update	The DAO ($60M), Curve ($73M)
Oracle Manipulation	Flash loan to distort price feed	Mango Markets ($116M)
Access Control	Missing permission checks	Ronin Bridge ($625M)
Flash Loan Attack	Borrow → manipulate → profit → repay	Beanstalk ($182M)
Logic Error	Flawed business logic	Wormhole ($326M)



(c) Joerg Osterrieder 2025-2026

Flash Loan Attack Anatomy

Attack Sequence (entire sequence in **ONE** transaction):

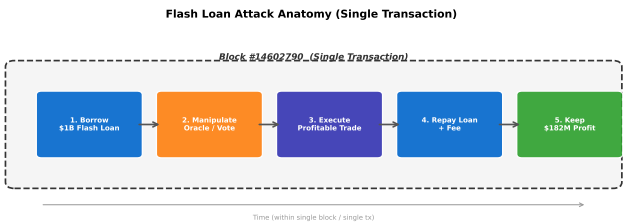
1. **Borrow** large sum—no collateral required (single-transaction loan)
2. **Manipulate** oracle price or governance vote
3. **Execute** profitable trade at manipulated price
4. **Repay** flash loan + fee
5. **Keep profit**—atomic execution means all-or-nothing

Case Study: Beanstalk Governance Attack (April 2022, $182M):

- Borrowed $1B via flash loan
- Voted on malicious governance proposal in same transaction
- Drained entire treasury before governance delay could activate

Prevention:

- **TWAP**: Time-weighted average prices resist single-block manipulation
- **Multi-block governance delays**: Prevent same-block vote + execution



Flash loans democratize capital access but also enable atomic exploits that were previously impossible

Code Is Law. But Who Writes the Law?

Smart contracts execute rules perfectly. But they cannot choose good rules. When the rules are wrong, billions get stolen.

What framework tells us which rules will work?

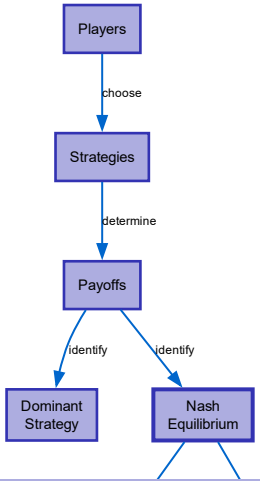
The Game

Your Game Theory Toolkit (Recap)

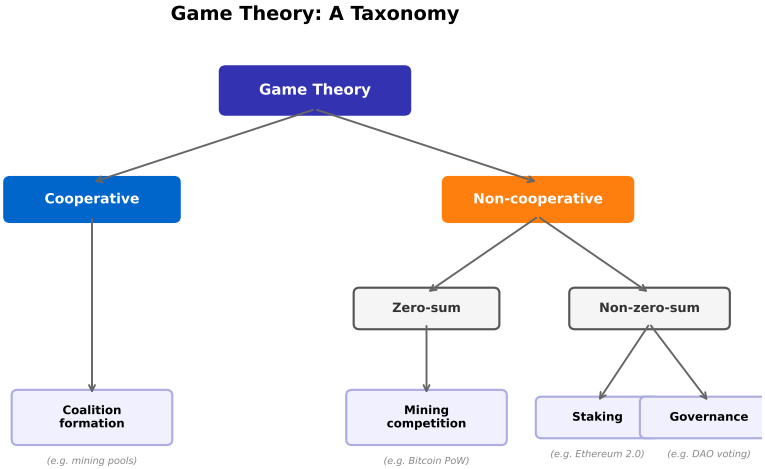
Recall from our intro lecture...

- **Players, Strategies, Payoffs** – the building blocks of every game
- **Dominant Strategy** – best move regardless of opponents
- **Nash Equilibrium** – no player gains by changing alone
- **Repeated Games** – cooperation can emerge when the game never ends

You already know the vocabulary. Now let's apply it to blockchain.

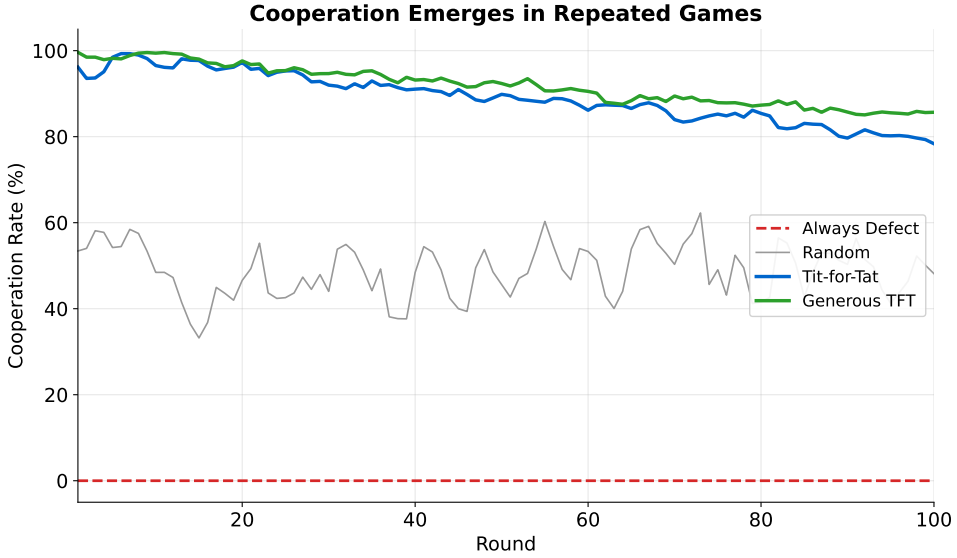


Types of Games



Most blockchain games are non-cooperative, non-zero-sum, and repeated

Repeated Games Change Everything



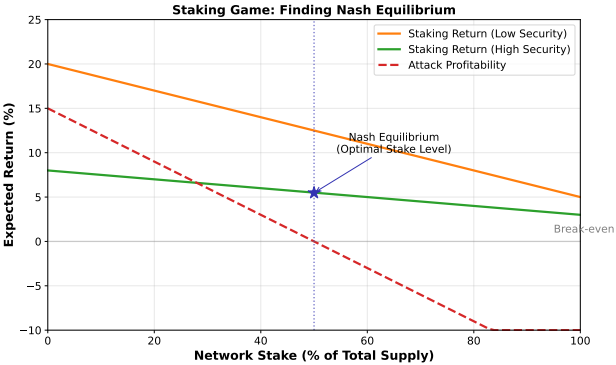
(c) Joerg Osterrieder 2025-2026

The Blockchain Is a Repeated Game

Every block is a round. Every validator is a player. Every transaction is a move.

What's the equilibrium?

Staking Game: Finding Equilibrium



Equilibrium stake level balances security and returns

Staking Game Analysis

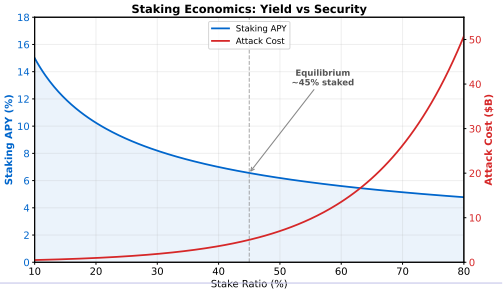
Players: Token holders deciding whether to stake.

Trade-offs:

- More stakers → more security, lower individual returns
- Fewer stakers → less security, higher individual returns
- Attacking becomes unprofitable as stake increases

Equilibrium:

- Stake level where returns equal opportunity cost
- Attack profitability becomes negative
- System is secure and sustainable



Staking rewards calibrated to achieve target stake ratio

Maximal Extractable Value (MEV)

When transaction ordering is a game, block builders are the players, and users are the pawns.

Definition: Profit extractable by reordering/inserting transactions.

MEV Examples:

- **Frontrunning**: Copy profitable trade, execute first
- **Sandwich attacks**: Surround user trade with own trades
- **Arbitrage**: Exploit price differences across DEXs
- **Liquidations**: Race to liquidate unhealthy positions

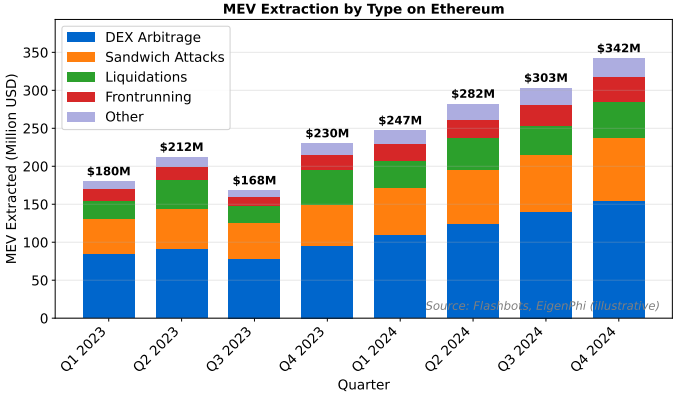
Game Theory Lens: MEV is a Nash equilibrium – block builders maximize profit by exploiting ordering. No individual builder benefits from unilaterally stopping.

Impact: Users pay hidden costs; validators extract value.

MEV tracker: `eigenphi.io` — **Research:** `writings.flashbots.net`

(c) Joerg Osterrieder 2025-2026

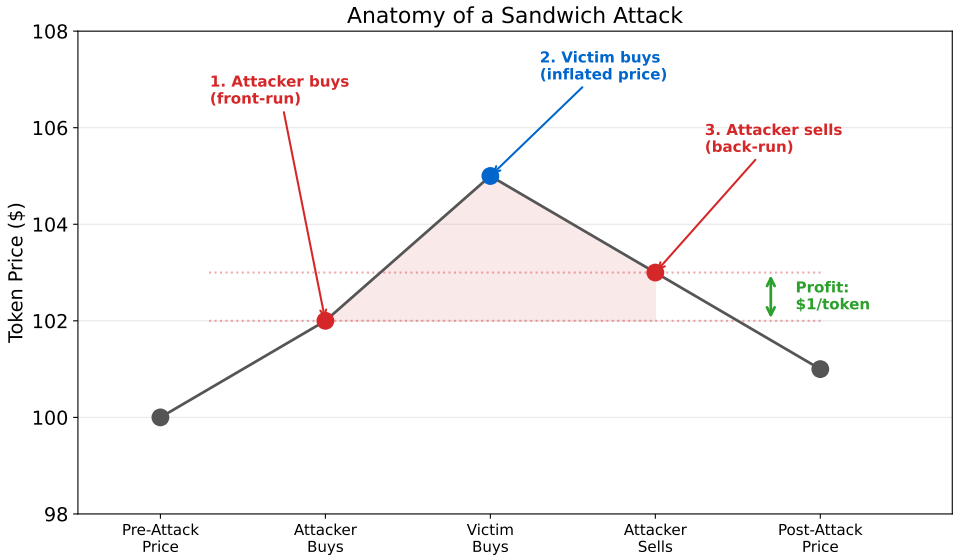
MEV Extraction by Type



DEX arbitrage dominates, but sandwich attacks growing — Source: Flashbots

(c) Joerg Osterrieder 2025-2026

Anatomy of a Sandwich Attack



MEV Mitigation Strategies

Protocol-Level:

- Flashbots: Private transaction pools
- Order-flow auctions: MEV goes to users
- Encrypted mempools: Hide transaction content

Application-Level:

- Batch auctions (CoW Protocol)
- Commit-reveal schemes
- Time-weighted average prices

MEV research: flashbots.net — ethereum.org/developers/docs/mev

(c) Joerg Osterrieder 2025-2026

Game Theory Reveals the Problem

Selfish actors exploit transaction ordering (MEV), governance tokens (flash loan attacks), and oracle feeds (price manipulation).

Can we design rules that harness selfishness instead?

The Design

What is Mechanism Design?

Definition: "Reverse game theory" – designing rules so self-interested players achieve desired outcomes.

The Goal:

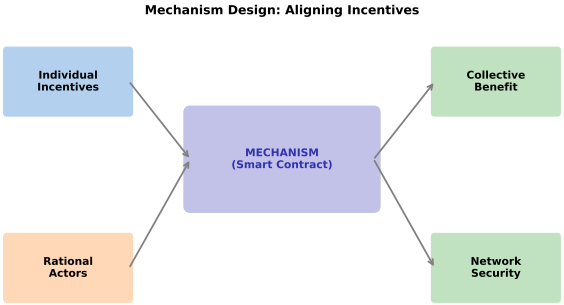
- Define desired outcome (security, fairness)
- Design rules/incentives to achieve it
- Ensure equilibrium matches desired outcome

In Crypto:

- Consensus mechanisms
- Token incentive structures
- Governance systems

Mechanism design: make doing the right thing profitable

Mechanism Design Flow



Key: Design rules so self-interest produces desired outcomes

Good mechanisms transform individual incentives into collective benefit

Incentive Compatibility

Definition: A mechanism where honest behavior is the best strategy.

Requirements:

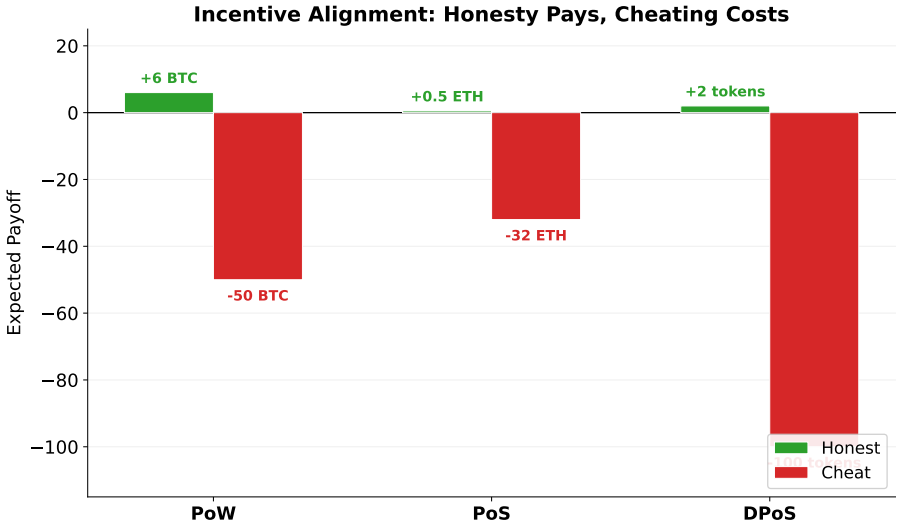
- Truth-telling is dominant strategy
- Cheating is punished (more than gained)
- Following rules is rewarded

Examples in Crypto:

- PoW: Mining honestly earns rewards
- PoS: Slashing makes attacks costly
- Oracles: Stake-weighted, penalize liars

Incentive-compatible systems don't rely on trust

Incentive Alignment Across Protocols



Well-designed mechanisms make cheating always have negative expected value

Bitcoin's Mechanism Design

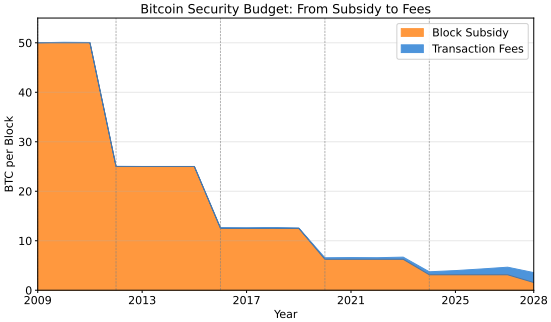
Desired Outcome: Secure, decentralized transaction ledger.

Mechanism:

- Block rewards for valid blocks
- Longest chain rule (follow majority)
- Difficulty adjustment (stable block time)
- Transaction fees (prioritization)

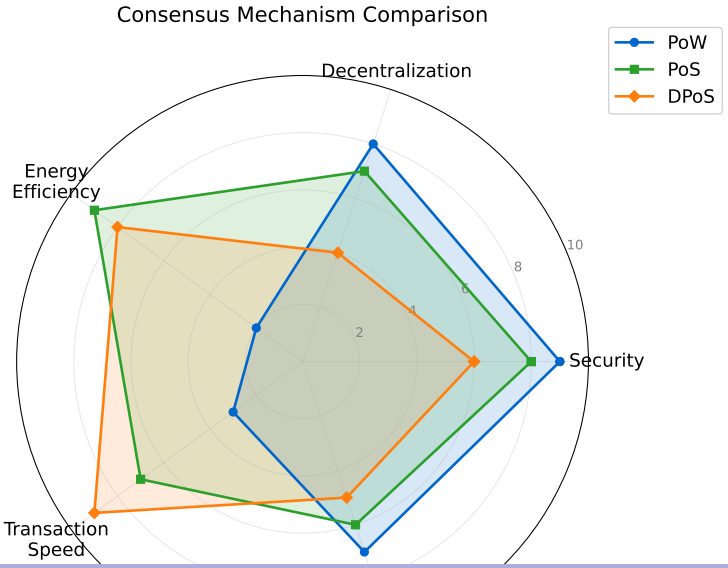
Why It Works:

- Honest mining is most profitable
- Attacking requires 51% hash power
- Attack destroys value of rewards



Bitcoin aligned miner incentives with network security

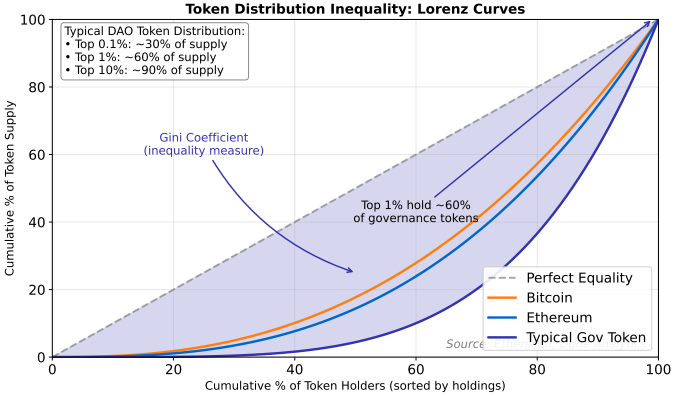
Comparing Consensus Mechanisms



Governing the Ungovernable

How do you make collective decisions when token holders are anonymous, unequal, and self-interested?

Token Holder Distribution



Governance tokens are highly concentrated—top 1% often hold 60%+ — Source: Etherscan

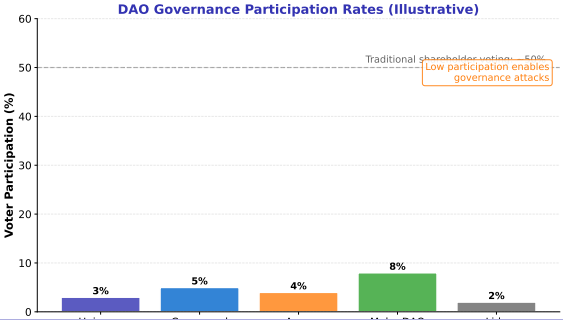
Governance Games

On-Chain Governance:

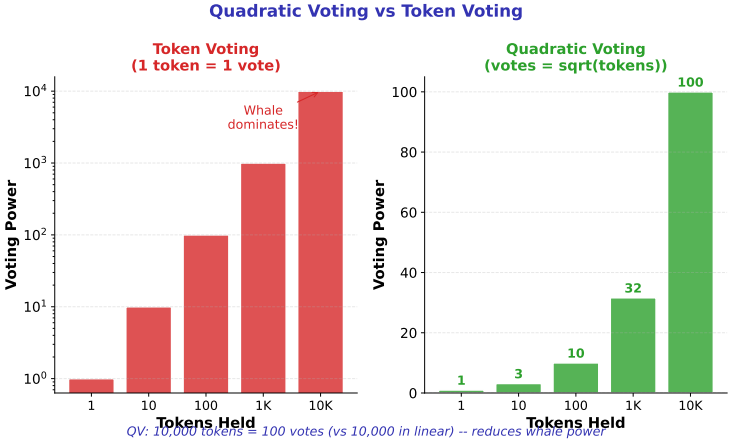
- Token voting on protocol changes
- Delegation to representatives
- Time-locked proposals

Game Theory Challenges:

- Voter apathy (rational ignorance)
- Plutocracy (whales dominate)
- Vote buying/bribery
- Short-term vs. long-term interests



Quadratic Voting: Fairer Governance?



Quadratic voting reduces whale dominance by making additional votes increasingly expensive

Quadratic Voting Mechanics

How It Works:

- Cost of n votes $= n^2$ tokens
- 1 vote $=$ 1 token, 10 votes $=$ 100 tokens
- Marginal cost increases with influence

Benefits:

- Reduces plutocracy (whales pay more per vote)
- Rewards conviction (multiple votes possible)
- More democratic outcomes

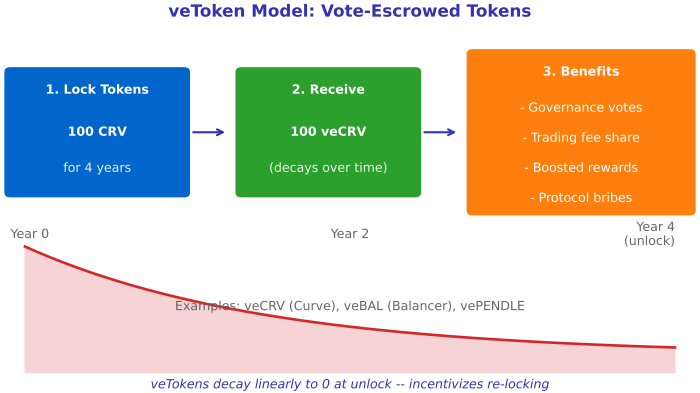
Challenges:

- Sybil attacks (split tokens across wallets)
- Requires identity/verification for full benefit

Gitcoin stats: gitcoin.co/program – **funded $67M+ to public goods**

(c) Joerg Osterrieder 2025-2026

veToken Model: Aligning Long-Term Incentives



Lock tokens longer = more voting power and rewards

(c) Joerg Osterrieder 2025-2026

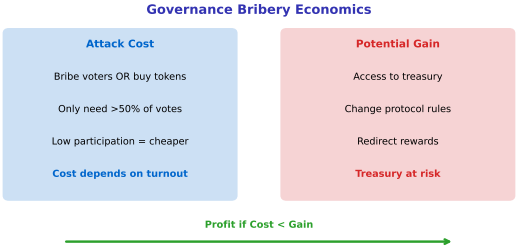
Governance Attacks

Flash Loan Governance Attack:

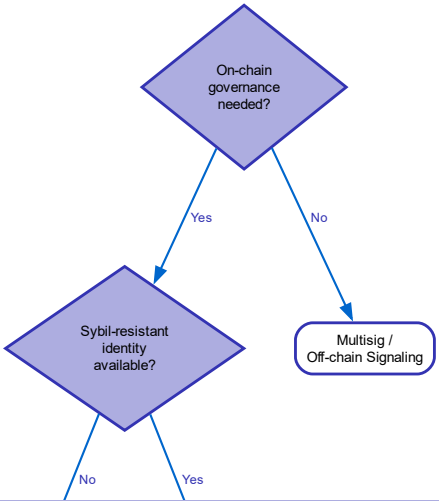
- Borrow governance tokens via flash loan
- Vote on malicious proposal in same transaction
- Return tokens—attack costs only gas
- Defense: Time-locked voting, snapshot balances

Vote Buying/Bribery:

- Platforms like Votium, Hidden Hand pay for votes
- "Bribes" redirect liquidity incentives
- Can be legitimate (CRV wars) or malicious



Choosing a Governance Mechanism



Game Theory: Commitment Devices

Traditional Commitment Devices:

- Odysseus and the Sirens: physically bound to the mast
- Burning bridges: eliminate retreat to force forward progress
- Public promises: reputation as collateral

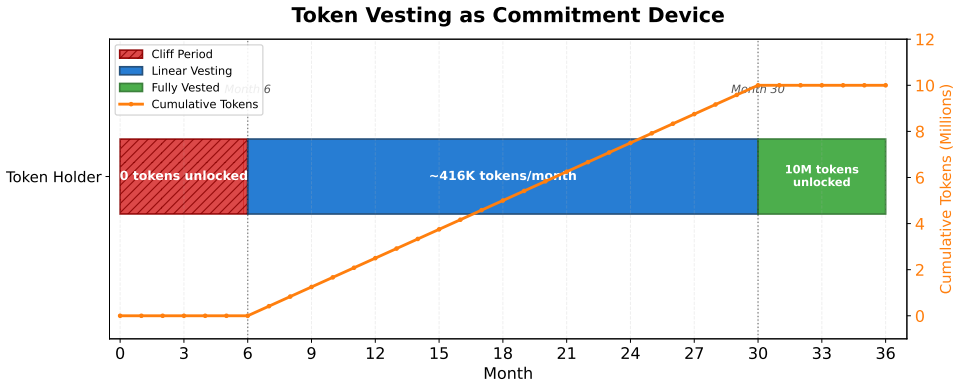
Smart Contract Commitment Devices:

- **Time-locks**: Funds locked until condition met—no early withdrawal possible
- **Vesting schedules**: Credible commitment to long-term project alignment
- **Staking as commitment**: Validators stake to prove honest behavior
- **Escrow contracts**: Trustless trade without intermediary

Key Insight: Smart contracts enable **trustless commitment**—the code enforces the commitment, not reputation or trust.

Blockchain turns theoretical commitment devices into enforceable code—no trust required

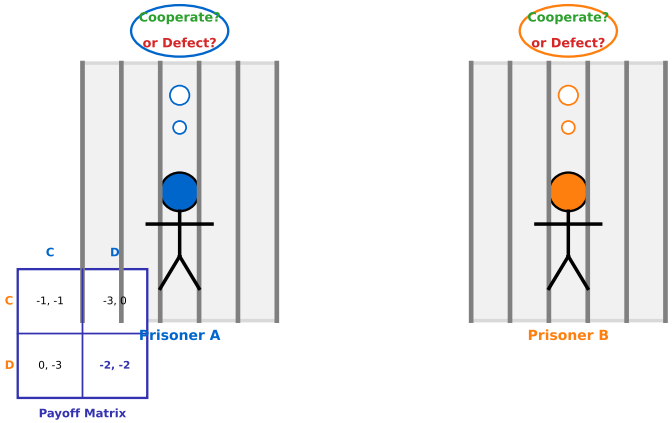
Token Vesting as Commitment



Vesting schedules are smart contract commitment devices — code enforces long-term alignment

(c) Joerg Osterrieder 2025-2026

Prisoner's Dilemma



Key Takeaways

What You Learned Today:

1. Smart contracts execute rules perfectly but cannot choose good rules
2. Game theory predicts how selfish actors behave under any set of rules
3. Mechanism design writes rules that make selfishness productive
4. Real crypto systems (PoS, QV, veTokens) are mechanism design in action

Core Insight: Crypto protocols do not work because participants are honest. They work because mechanism design makes honesty the most profitable strategy.

Next lesson: Regulation, Risks, and Future Trends

Questions for Reflection

1. Why can't smart contracts alone solve the trust problem?
2. How does MEV demonstrate a Nash equilibrium that harms users?
3. What makes a governance mechanism incentive-compatible?
4. If you were designing a new DAO, which governance mechanism would you choose and why?

Discussion: Crypto protocols harness selfishness by design. Where does this approach break down?

Prepare for next session

Thank You

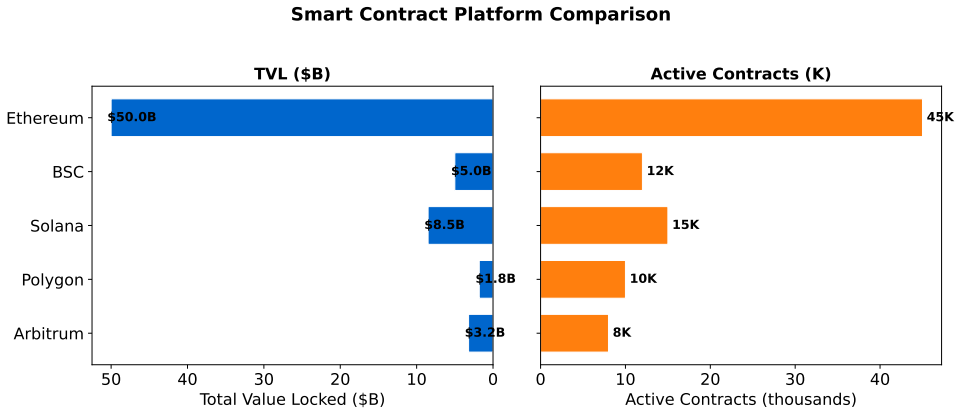
Questions?

Course materials: digital-ai-finance.github.io/crypto-economics

Appendix: Advanced Topics

Appendix: Smart Contract Reference

Smart Contract Ecosystem



Ethereum dominates by TVL; multichain ecosystem growing rapidly

(c) Joerg Osterrieder 2025-2026

Smart Contract Languages

Solidity:

- Primary programming language for Ethereum smart contracts
- JavaScript-like syntax, statically typed
- Compiles to **EVM** bytecode
- Example: `function transfer(address to, uint amount) public {}`

Other Languages:

- **Vyper**: Python-like, security-focused
- **Rust**: Used for Solana smart contracts
- **Move**: Used for Aptos and Sui blockchains

Gas: Unit measuring computational effort. Users pay gas fees to execute transactions.

Learn Solidity: docs.soliditylang.org — **Gas tracker:** etherscan.io/gastracker

Token Standards

ERC-20: Fungible Tokens

- Standard for interchangeable tokens (USDC, DAI, UNI)
- Required functions: `transfer()`, `balanceOf()`, `approve()`, `transferFrom()`
- Enables wallet and exchange compatibility

ERC-721: Non-Fungible Tokens (NFTs)

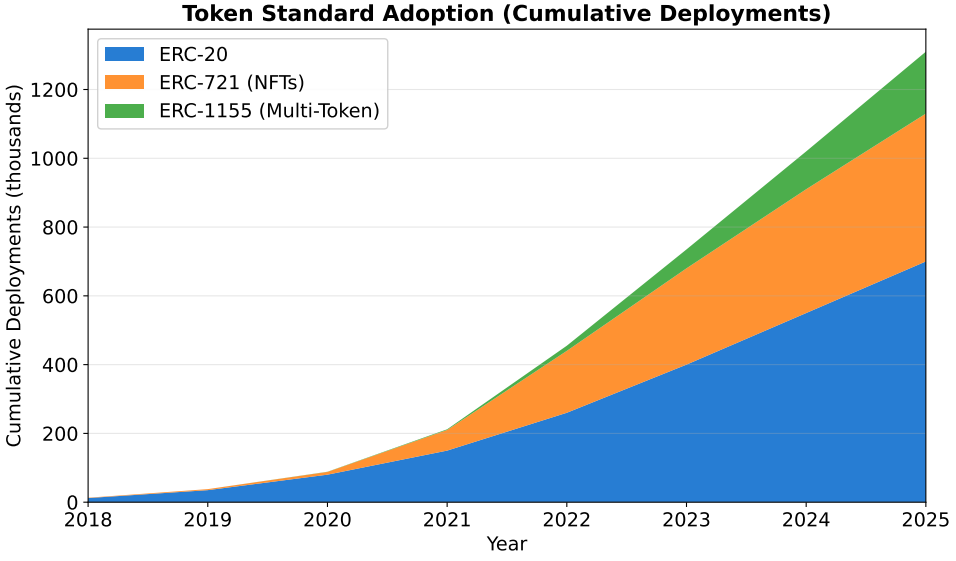
- Standard for unique, non-fungible tokens
- Each token has unique ID and metadata
- Used for digital art, collectibles, game items
- Functions: `ownerOf()`, `safeTransferFrom()`

Other Standards: ERC-1155 (multi-token), ERC-4626 (tokenized vaults)

ERC standards: `eips.ethereum.org` — **OpenZeppelin reference:** `docs.openzeppelin.com`

(c) Joerg Osterrieder 2025-2026

Token Standard Growth



(c) Joerg Osterrieder 2025-2026

Account Abstraction (ERC-4337)

Problem: Externally Owned Accounts (EOAs) are limited:

- No transaction batching—one operation per transaction
- No social recovery—seed phrase is single point of failure
- No gas sponsorship—users must hold ETH to transact
- Only ECDSA signatures—no alternative authentication

Solution: ERC-4337 introduces `UserOperations` processed by **Bundlers**.

- Smart contract wallets replace EOAs as primary accounts
- **Gas sponsorship**: Paymasters cover gas for users
- **Session keys**: Temporary permissions without full wallet access
- **Multi-sig**: Built-in without external contracts
- **Arbitrary verification logic**: Biometrics, multi-factor, passkeys

---

ERC-4337 makes smart contracts first-class citizens without changing the Ethereum protocol

Account Abstraction Architecture

Transaction Flow:

EOA Wallet vs. Smart Account:

Feature	EOA	Smart Account
Recovery	Seed phrase only	Social recovery, guardians
Batching	One tx at a time	Multiple operations
Gas payment	Must hold ETH	Paymaster sponsors
Authentication	ECDSA only	Any signature scheme
Upgradeable	No	Yes (proxy pattern)

Why It Matters: Account abstraction removes the largest UX barrier to mainstream adoption—users no longer need to manage private keys or hold ETH before their first interaction.

Account abstraction adoption grew 10x in 2024, driven by gasless onboarding UX

(c) Joerg Osterrieder 2025-2026

Contract Upgradeability

The Challenge:

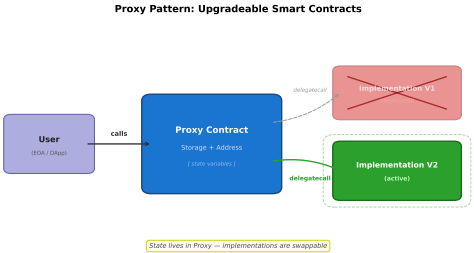
- Smart contract code is immutable once deployed
- Bugs cannot be fixed, features cannot be added
- Users interact with fixed address

Proxy Pattern Solution:

- **Proxy contract**: Stores state, users interact with this address
- **Logic contract**: Contains the actual code
- Proxy delegates calls to logic contract using `delegatecall`
- Logic contract can be swapped to "upgrade" functionality
- State remains in proxy, preserved across upgrades

Trade-off:

- **Pro**: Flexibility to fix bugs and add features
- **Con**: Introduces centralization risk (who controls upgrades?)



Proxy patterns: OpenZeppelin upgradeable contracts — docs.openzeppelin.com/upgrades

Appendix: Game Theory Reference

Introduction to Game Theory

Definition: Mathematical study of strategic decision-making.

Key Concepts:

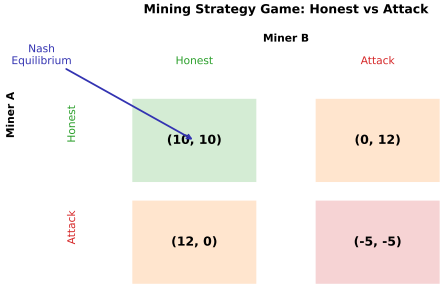
- **Players**: Decision makers (miners, validators, users)
- **Strategies**: Available actions for each player
- **Payoffs**: Outcomes/rewards for strategy combinations
- **Equilibrium**: Stable state where no one wants to change

Why It Matters: Crypto protocols must work when everyone acts selfishly.

Game theory ensures protocols are robust against selfish behavior

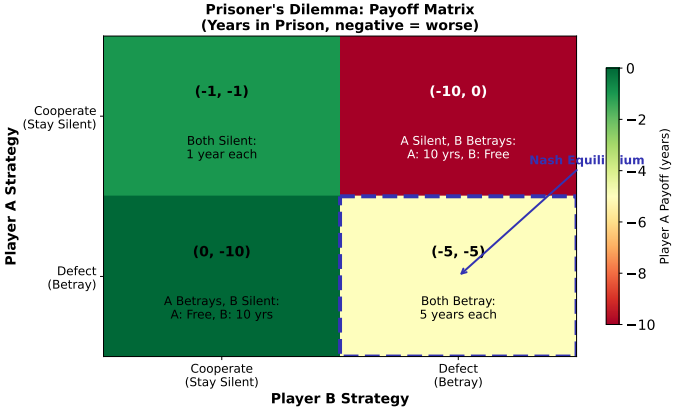
(c) Joerg Osterrieder 2025-2026

The Prisoner's Dilemma



Individual rationality leads to collective suboptimality

Prisoner's Dilemma: Payoff Matrix



Nash equilibrium emerges even when mutual cooperation is better for both

Prisoner's Dilemma Analysis

The Setup:

- Two prisoners, cannot communicate
- Each can cooperate (stay silent) or defect (confess)
- Payoffs depend on both choices

The Dilemma:

- Best collective outcome: Both cooperate
- Dominant strategy: Defect (regardless of other's choice)
- Nash equilibrium: Both defect (worse for both)

Crypto Parallel: Why would miners/validators cooperate?

Good protocols make cooperation the dominant strategy

Nash Equilibrium

Definition: A state where no player can improve their outcome by unilaterally changing strategy.

Formal Definition:

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

- s_i^* = player i's equilibrium strategy
- s_{-i}^* = other players' equilibrium strategies
- u_i = player i's utility/payoff

Implication: At Nash equilibrium, system is stable.

Well-designed protocols have desirable Nash equilibria

Appendix: Governance Reference

Auction Mechanisms

Common Crypto Auctions:

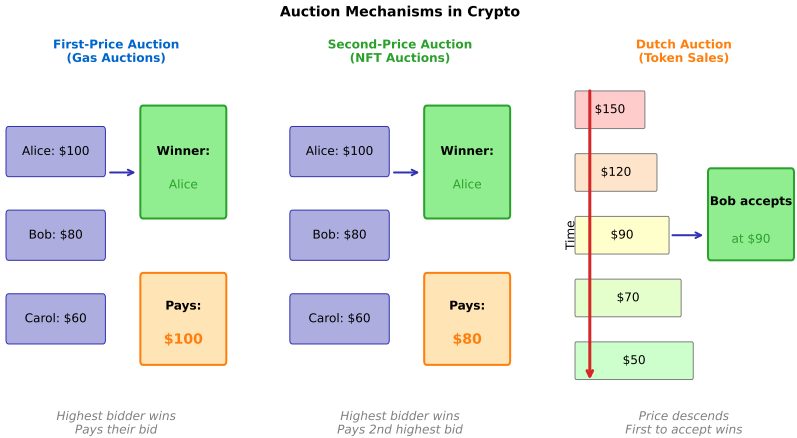
- Gas auctions (transaction priority)
- NFT auctions (English, Dutch)
- Liquidation auctions
- Token sales (bonding curves)

EIP-1559 Innovation:

- Base fee burned (not to miners)
- Priority fee to validators
- More predictable **gas prices**
- Reduced bidding wars

Auction design significantly impacts user experience

Auction Mechanisms Compared



Different auctions suit different use cases: gas (first-price), NFTs (second-price), token sales (Dutch)

(c) Joerg Osterrieder 2025-2026

Schelling Points

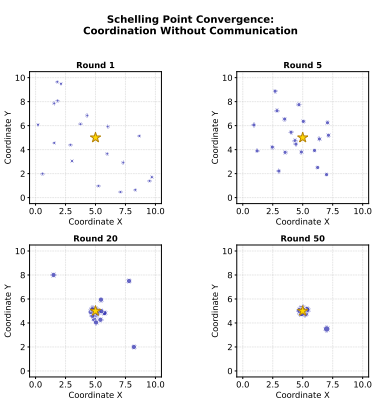
Definition: Focal points where people naturally coordinate without communication.

Use in Crypto:

- Oracle networks (report true price)
- Prediction markets (report true outcome)
- Dispute resolution (vote for truth)

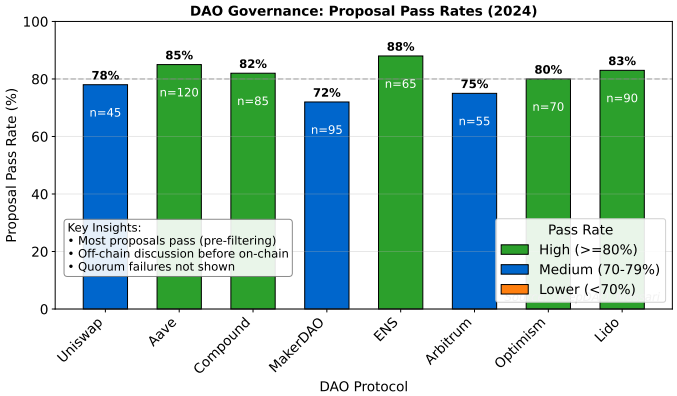
Example – Kleros:

- Jurors stake tokens
- Vote on disputes
- Rewards for voting with majority
- Penalty for minority votes



Schelling points make truth-telling focal

DAO Proposal Outcomes



High pass rates reflect pre-filtering through off-chain discussion — Source: DeepDAO

(c) Joerg Osterrieder 2025-2026

Common Knowledge and On-Chain Transparency

Definition: Common knowledge = everyone knows X, everyone knows that everyone knows X, ad infinitum.

Traditional Problem: Hard to establish common knowledge without a trusted authority.

Blockchain as Common Knowledge Generator:

- All transactions **publicly visible** to every participant
- All state **verifiable** by anyone running a node
- No information asymmetry about on-chain state

Implications for Mechanism Design:

- Enables mechanisms relying on public verifiability (e.g., quadratic funding)
- Eliminates certain types of **adverse selection**
- Creates new equilibria impossible in private-information settings

---

Public blockchains solve the common knowledge problem that limits many real-world mechanisms

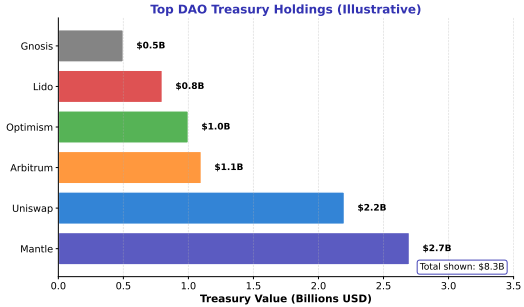
Case Study: The Curve Wars

Background:

- Curve Finance: Largest stablecoin DEX
- CRV token directs liquidity incentives
- veCRV holders vote on "gauge weights"

The War:

- Protocols compete for CRV emissions to their pools
- Convex accumulates veCRV to control votes
- Bribes paid to veCRV holders for favorable votes via Votium, Hidden Hand



Outcome:

Created "yield-as-governance" paradigm—protocols pay for liquidity via governance.

Bribe data: `llama.airforce` **(Llama Airforce dashboard)**

Technical: EIP-1559 Mechanism Design

Before EIP-1559:

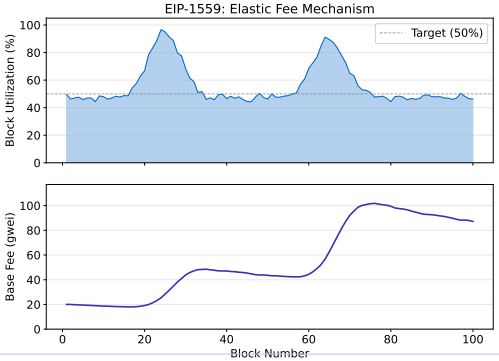
- First-price auction for gas
- Unpredictable fees
- Users overpay to ensure inclusion

After EIP-1559:

- Base fee: Algorithmically determined, burned
- Priority fee: Tip to validator
- Base fee adjusts to target 50% block fullness

Mechanism Properties:

- More predictable fees
- Reduced manipulation (can't pay for empty blocks)
- ETH becomes deflationary during high usage



ETH burn: ultrasound.money — **EIP spec:** eips.ethereum.org/EIPS/eip-1559

Advanced: Futarchy (Prediction Market Governance)

Concept: "Vote on values, bet on beliefs" (Robin Hanson)

How It Works:

1. Define success metric (e.g., token price, TVL)
2. Create prediction markets for each proposal
3. Market prices reveal expected outcomes
4. Implement proposal with best predicted outcome

Benefits:

- Aggregates distributed information
- Incentivizes informed participation
- Reduces influence of uninformed voters

Projects Exploring: Gnosis, MetaDAO (Solana)

Futarchy: `metadao.fi` **(Solana)** — `gnosis.io`

(c) Joerg Osterrieder 2025-2026

Technical: Formal Verification of Smart Contracts

What is Formal Verification?
Mathematical proof that code behaves as specified—not just testing.

Methods:

- **Model checking**: Exhaustively verify all states
- **Theorem proving**: Mathematical proofs of correctness
- **Symbolic execution**: Explore all execution paths

Tools:

- Certora Prover (Aave, Compound audits)
- Mythril, Slither (vulnerability detection)
- K Framework (formal semantics)

Limitation: Can only verify code matches spec—spec itself may be wrong.

Tools: certora.com — github.com/Consensys/mythril

(c) Joerg Osterrieder 2025-2026