L07 — Smart Contracts & Game Theory

Cheatsheet

Smart Contract Basics

Definition

Self-executing code deployed on a blockchain that runs automatically when pre-defined conditions are met — no intermediary required.

Key properties:

- **Deterministic** — same inputs always produce same outputs
- **Immutable** — code cannot be changed after deployment (proxy patterns enable upgrades at the cost of centralization)
- **Transparent** — code and state publicly visible on-chain
- **Trustless** — executes without relying on any third party

Limitations: cannot access external data (need oracles), cannot self-trigger, cannot reverse confirmed transactions.

Solidity — primary language for Ethereum; JavaScript-like syntax, statically typed, compiles to EVM bytecode.

```
pragma solidity ^0.8.0;
```

Token standards:

- ERC-20: fungible tokens (USDC, UNI, AAVE)
- ERC-721: non-fungible tokens (NFTs)
- ERC-1155: multi-token (fungible + NFT in one contract)

Game Theory Fundamentals

Core Components

Players — rational decision-makers (miners, validators, users).
Strategies — set of available actions for each player.
Payoffs — outcomes (rewards/penalties) for each strategy profile.
Information — what each player knows when choosing.

Dominant strategy — a strategy that yields the highest payoff *regardless* of what other players do.

Cooperative vs. non-cooperative — whether binding agreements can be made; most blockchain settings are non-cooperative.

Zero-sum vs. non-zero-sum — MEV is often zero-sum (validator gains = user loss); protocol design aims for positive-sum outcomes.

Repeated games — cooperation becomes easier when players interact repeatedly; reputation matters.

Nash Equilibrium

Definition

A strategy profile $(s_1^*, \ldots, s_n^*)$ such that no player i can increase their payoff by unilaterally deviating:

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

s_{-i}^* denotes all *other* players' equilibrium strategies.

Key insight: at a Nash equilibrium the system is *stable* — no individual has incentive to change behaviour.

Crypto examples:

- Bitcoin mining: honest mining is the Nash equilibrium because attacking destroys the value of block rewards.
- PoS staking: validators stake until marginal reward equals opportunity cost; equilibrium stake ratio secures the chain.

Well-designed protocols have desirable Nash equilibria (honest behaviour = best response).

Prisoner's Dilemma

Setup: Two players each choose *Cooperate* (C) or *Defect* (D); payoffs below (row player, column player):

	C	**D**
C	$(-1, -1)$	$(-10, 0)$
D	$(0, -10)$	$(-5, -5)$

Result: D is a dominant strategy ⇒ Nash equilibrium is (D, D), even though (C, C) is Pareto-superior.

Blockchain parallel (mining): each miner can cooperate (follow protocol) or defect (mine selfish blocks, censor transactions). PoW makes defection costly: honest mining is the dominant strategy when attack cost exceeds expected gain.

Mechanism Design

Definition

"Reverse game theory": *design* the rules of the game so that self-interested players, acting at their own Nash equilibrium, achieve the **designer's desired outcome**.

Incentive compatibility (IC): truth-telling / honest behaviour is the dominant strategy. Achieved when cheating is penalised more than the gain (slashing in PoS, hash-power cost in PoW).

Individual rationality (IR): participation is voluntary only if expected payoff ≥ outside option.

Revelation principle: any outcome achievable by a complex mechanism is also achievable by a *direct revelation mechanism* where players truthfully report their types.

In crypto: PoW block rewards (honest mining = profit), PoS slashing (attack = loss), oracle staking (accurate data = reward).

Quadratic Voting (QV)

Cost Function

Cost of casting n votes $= n^2$ tokens.
1 vote costs 1 token; 10 votes cost 100 tokens.
Marginal cost of influence rises quadratically.

Why it reduces whale dominance: a wallet with $10\times$ more tokens can cast only $\sqrt{10} \approx 3.16\times$ as many votes.

Benefits:

- More democratic outcome weighting
- Rewards conviction (multiple votes on issues you care about)

Vulnerability: Sybil attacks — split tokens across many wallets to buy votes cheaply; mitigated by identity verification (e.g., Gitcoin Passport).

Quadratic funding extends QV to public goods: matching funds proportional to $(\sum \sqrt{c_i})^2$.

veToken Model (Vote-Escrowed)

Mechanism: lock governance token for 1–4 years ⇒ receive *veTokens* proportional to amount × duration.

veToken voting power decays linearly to zero at lock expiry. Re-locking restores power.

Benefits for holders: boosted yield, protocol fee share, vote on liquidity gauge weights.

Protocol benefits: reduces circulating supply, aligns holders with long-term health, creates stickiness.

Example (Curve Finance): lock CRV → veCRV → vote on pool incentives ("Curve Wars"); third parties (Convex, Votium) accumulate veCRV and sell vote influence.

MEV & Front-running

MEV (Maximal Extractable Value)

Profit extractable by a block producer through **reordering**, **inserting**, or **censoring** transactions within a block.

MEV strategies:

- **Front-running**: copy profitable tx, submit with higher gas to execute first.
- **Sandwich attack**: place buy *before* and sell *after* victim's large swap ⇒ profit from induced slippage.
- **DEX arbitrage**: exploit price discrepancies across pools.
- **Liquidations**: race to liquidate under-collateralised positions for the 5–10% bonus.

Game-theoretic framing: MEV is a commons problem — validators and searchers compete; users bear hidden costs.

Mitigations: Flashbots private mempools, order-flow auctions, encrypted mempools, batch auctions (CoW Protocol).

Auction Mechanisms

Type	Description
English	Ascending bids; highest wins; common for NFTs
Dutch	Descending price; first bidder wins; used in token sales
Sealed-bid	Simultaneous bids; highest wins; first-price
Vickrey	Sealed-bid; highest wins but pays *second-highest* price; truth-telling dominant

EIP-1559 (gas auctions): base fee burned + priority tip to validator; more predictable than first-price; ETH becomes deflationary at high usage.

Revenue equivalence theorem: under symmetric, independent private values, all four standard auction formats yield the same expected revenue to the seller.

DAO Governance

On-chain governance flow:

1. Off-chain discussion (Snapshot, forum)
2. On-chain proposal submitted
3. Voting period (token-weighted)
4. Quorum threshold must be met
5. Timelock delay before execution (e.g., 48 h)
6. Multisig or automated execution

Game-theory challenges:

- *Rational apathy*: cost of informed voting > individual impact ⇒ low participation.
- *Plutocracy*: whales dominate token-weighted votes.
- *Vote buying*: platforms (Votium, Hidden Hand) pay for votes.
- *Flash-loan governance attack*: borrow tokens, vote on malicious proposal, repay — all in one transaction. Defense: snapshot balances at block $N - k$, timelocks.

Example: Beanstalk (Apr 2022) flash-loan governance attack — $182M stolen.

Smart Contract Vulnerabilities

Reentrancy: attacker's contract calls victim → victim sends ETH before updating state → attacker re-enters and drains funds. *Prevention*: Checks-Effects-Interactions pattern; mutex locks. *Example*: The DAO hack (2016), $60M.

Integer overflow/underflow: arithmetic wraps around type limits. *Prevention*: Solidity ≥0.8 reverts on overflow by default; use SafeMath for older versions.

Flash loan attacks: uncollateralised capital used to manipulate prices or governance in a single transaction. *Defense*: TWAP oracles, governance snapshots, multi-block delays.

Access control bugs: functions callable by unintended addresses. *Prevention*: `onlyOwner` / role-based access (OpenZeppelin `AccessControl`).

Oracle manipulation: corrupt price feed ⇒ trigger mispriced liquidations or minting.

Proxy storage collision: delegatecall-based upgrades can overwrite storage slots. *Prevention*: EIP-1967 standardised storage slots.