

Cryptoeconomics of Defifa

Claude Opus 4.6 (Anthropic) in coordination with Jango from the Defifa Team.

This analysis was generated by Claude Opus 4.6 based on its study of the Defifa V5 codebase and the [Revnet Whitepaper](#) by CryptoEconLab.

March 2026

Abstract

Defifa is a prediction-game protocol built on Juicebox V5 that transforms NFT minting into a parimutuel wagering mechanism with governance-ratified outcomes. Players purchase ERC-721 game pieces representing competing tiers (teams, candidates, outcomes), forming a shared treasury. After the event concludes, a decentralized attestation process ratifies a scorecard that assigns weights to each tier, redistributing the treasury proportionally. This paper formalizes the cryptoeconomic mechanics of Defifa games: the prize distribution formula, the attestation governance model, the fee extraction pipeline, the protocol-token incentive layer, and the rational actor strategies that emerge. We derive solvency guarantees, characterize equilibrium behavior under various participation profiles, analyze the game-theoretic properties of the scorecard ratification process, and identify the parameter regimes that maximize game integrity and participant welfare.

Contents

1	Introduction	3
1.1	What is Defifa?	3
1.2	How a Defifa Game Works (at a glance)	3
1.3	The Design Parameters	3
2	Mathematical Model of Defifa Economics	4
2.1	Parameters and State Variables	4
2.2	Minting — Pot Formation	4
2.3	Refund — Optionality Window	5
2.4	Prize Distribution — The Scorecard Formula	5
2.5	Fee Extraction Pipeline	6
2.6	Protocol Token Allocation	7
3	Attestation Governance and Scorecard Ratification	7
3.1	Voting Power Model	7
3.2	Quorum and Ratification Conditions	7
3.3	Scorecard Lifecycle	8
3.4	Resistance to Strategic Manipulation	8
4	Price Dynamics and Value Flows	9
4.1	NFT Intrinsic Value During Minting	9
4.2	Post-Scorecard Valuation	9
4.3	Secondary Market Implications	10

5	Rational Actor Analysis	10
5.1	Mint-Phase Strategy: Entry Timing	10
5.2	Refund-Phase Strategy: Option Exercise	10
5.3	Scoring-Phase Strategy: Attestation Delegation	11
5.4	Complete-Phase Strategy: Claim vs Hold	11
6	Solvency and Conservation Laws	11
6.1	The Conservation Guarantee	11
6.2	Solvency Under Sequential Cash-Outs	11
6.3	Fee Impact on Total Claimable Value	12
7	Game-Theoretic Properties	12
7.1	Defifa as a Parimutuel Mechanism	12
7.2	Information Aggregation	12
7.3	Multi-Game Dynamics and Protocol Flywheel	12
8	Parameter Design Space	13
8.1	Tier Count and Price Calibration	13
8.2	Timing Parameters	13
8.3	Fee Calibration and Protocol Sustainability	13
9	Open Problems and Mechanism Design Recommendations	14
9.1	Governance Deadlock and Fund Recovery: A Deep Study	14
9.1.1	Historical Context	14
9.1.2	Exhaustive Deadlock Scenario Analysis	14
9.1.3	Effectiveness of the Default Attestation Delegate	15
9.1.4	Candidate Mechanism A: Minimum Participation Threshold	15
9.1.5	Candidate Mechanism B: Scorecard Ratification Timeout	15
9.1.6	Do We Need a Formal State?	15
9.1.7	Assessment and Recommendation	15
9.2	Cheap Cross-Tier Attestation Capture	16
9.3	Prize Pool Under-Allocation	16
9.4	Attestation Timing Misconfiguration	16
9.5	Pre-Scoring Scorecard Submission	17
9.6	Fee Extraction Fragility	17
10	Conclusions and Practical Implications	17

1 Introduction

1.1 What is Defifa?

Defifa is a prediction-game protocol that transforms the act of purchasing an NFT into a wager on the outcome of a real-world event. It is deployed using the Juicebox V5 protocol and governed by a combination of immutable smart-contract rules and a minimal, time-bounded governance process for outcome resolution.

A Defifa game is a *tokenized parimutuel pool*: money goes in via NFT purchases, forming a shared pot; after the event concludes, a governance process assigns weights to each tier (team, outcome, candidate), and the pot is distributed proportionally. The game pieces are ERC-721 tokens organized into tiers, where each tier represents a distinct prediction. The purchase price of a tier token is fixed at game creation, and the payout is determined by post-event scorecard ratification.

Defifa games are:

- **Deterministic in structure**: all phases, durations, tier prices, and fee schedules are fixed at deployment.
- **Governance-minimal**: the only human input is the scorecard—a mapping from tiers to weights—ratified through an attestation process.
- **Self-custodial**: all funds remain in the Juicebox treasury; no operator can access them outside the protocol rules.
- **Composable**: games are standard Juicebox projects, inheriting the full protocol’s accounting, terminal, and hook infrastructure.

1.2 How a Defifa Game Works (at a glance)

1. **Mint (pot formation)**. During the mint phase, anyone can purchase NFTs representing tiers. Each NFT has a fixed price denominated in the game’s base asset (e.g., ETH). All payments flow into a shared treasury—the *pot*. Players may delegate their attestation power to a chosen delegate at mint time.
2. **Refund (optional exit window)**. If configured, a refund phase follows minting. During this period, players may burn their NFTs to reclaim the original mint price, allowing a risk-free exit for those who change their minds. No new mints are accepted.
3. **Score (outcome resolution)**. Once the real-world event concludes, anyone may propose a *scorecard*—a vector of weights summing to $W_{\text{total}} = 10^{18}$ —assigning each tier its share of the pot. NFT holders attest to the scorecard they believe reflects the correct outcome. Once a scorecard achieves quorum, it can be ratified.
4. **Complete (prize distribution)**. After ratification, protocol fees are extracted, and the remaining pot is available for claims. Each NFT holder burns their token to receive their proportional share, plus any accrued protocol tokens (\$DEFIFA and \$BASE_PROTOCOL).

1.3 The Design Parameters

A Defifa game is fully specified at deployment by a parameter tuple:

$$\mathcal{G} = \left(\{T_i\}_{i=1}^N, t_{\text{mint}}, t_{\text{refund}}, t_{\text{start}}, \phi_{\text{defifa}}, \phi_{\text{base}}, \mathcal{S}, \tau_{\text{attest}}, \tau_{\text{grace}} \right) \quad (1)$$

Where:

1. **Tier configuration** $\{T_i\}_{i=1}^N$: For each of the N tiers, a fixed price p_i , an optional reserved rate ρ_i , and a reserved-token beneficiary address. The initial supply per tier is set to 999,999,999 (effectively unlimited).
2. **Mint period duration** (t_{mint}): How long the minting window stays open, in seconds.

3. **Refund period duration** (t_{refund}): How long the refund window stays open after minting closes. May be zero (no refund phase).
4. **Game start time** (t_{start}): When the scoring phase begins—typically aligned with the real-world event’s conclusion.
5. **Defifa fee divisor** (ϕ_{defifa}): The fraction $1/\phi_{\text{defifa}}$ of the pot sent to the Defifa protocol project. Default: $\phi_{\text{defifa}} = 20$ (5%).
6. **Base protocol fee divisor** (ϕ_{base}): The fraction $1/\phi_{\text{base}}$ of the pot sent to the base protocol project. Default: $\phi_{\text{base}} = 20$ (5%).
7. **Splits** (\mathcal{S}): Additional payout splits configured at deployment (e.g., for game organizers, charities).
8. **Attestation start time** (τ_{attest}): Delay before attestation voting opens on a submitted scorecard.
9. **Attestation grace period** (τ_{grace}): Duration of the attestation voting window.

Once set, the tuple \mathcal{G} is immutable. Phase transitions occur automatically by timestamp, with the scoring phase having infinite duration (duration = 0) until the scorecard is ratified.

2 Mathematical Model of Defifa Economics

2.1 Parameters and State Variables

The economic behavior of a Defifa game is determined jointly by:

1. The immutable game parameters \mathcal{G} (cf. Section 1.3), fixed at deployment;
2. The evolving state variables, which track the pot, token supplies, and claim status over time.

State variables. The core dynamic variables are listed in Table 1.

Variable	Description
$B(t)$	Pot (treasury balance) at time t
$n_i(t)$	Number of NFTs minted in tier i at time t
$N_{\text{total}}(t)$	Total NFTs outstanding: $\sum_i n_i(t)$
$M(t)$	Total mint cost accumulated: $\sum_i n_i(t) \cdot p_i$
w_i	Scorecard weight assigned to tier i ($\sum_i w_i = W_{\text{total}}$)
$d_i(t)$	Tokens redeemed from tier i after ratification
B_{prize}	Net prize pool after fee extraction

Table 1: Core state variables of a Defifa game.

At any time t , the state of the game is fully determined by the pair $(\mathcal{G}, \{B(t), n_i(t), w_i, d_i(t)\})$, where \mathcal{G} is the fixed game configuration and the second component evolves endogenously as players interact.

2.2 Minting — Pot Formation

During the mint phase $[t_{\text{mint_start}}, t_{\text{mint_start}} + t_{\text{mint}})$, any participant may purchase NFTs from any tier i at the fixed price p_i per token.

Minted quantity. For a payment amount x of base asset directed at tier i :

$$q_i = \left\lfloor \frac{x}{p_i} \right\rfloor \quad (2)$$

Reserved minting. If tier i has a reserved rate $\rho_i > 0$, then for every ρ_i tokens minted by paying players, one additional token is minted to the reserved-token beneficiary. Reserved

tokens are *not* paid for, but their cost is counted toward $M(t)$ for purposes of protocol-token distribution (cf. Section 2.6).

State updates. At the instant of a mint event where player j purchases q tokens of tier i :

$$\begin{aligned} B(t^+) &= B(t^-) + q \cdot p_i && \text{(Treasury balance)} \\ n_i(t^+) &= n_i(t^-) + q && \text{(Tier supply)} \\ M(t^+) &= M(t^-) + q \cdot p_i && \text{(Total mint cost)} \end{aligned}$$

Pot composition. At the end of the mint phase, the pot is:

$$B_{\text{mint}} = \sum_{i=1}^N n_i \cdot p_i \quad (3)$$

This is the total capital at risk in the game, and represents the complete prize pool before fee extraction.

2.3 Refund — Optionality Window

If $t_{\text{refund}} > 0$, a refund phase follows minting. During $[t_{\text{mint_end}}, t_{\text{mint_end}} + t_{\text{refund}})$:

- No new mints are accepted (**pausePay = true**).
- Any NFT holder may burn their token to reclaim its mint price.

Refund mechanics. A player burning q tokens of tier i receives exactly $q \cdot p_i$ base asset from the treasury:

$$R_{\text{refund}} = q \cdot p_i \quad (4)$$

State updates. After a refund:

$$B(t^+) = B(t^-) - q \cdot p_i \quad (5)$$

$$n_i(t^+) = n_i(t^-) - q \quad (6)$$

$$M(t^+) = M(t^-) - q \cdot p_i \quad (7)$$

The refund phase creates a *free option* for participants: they can observe late-breaking information (injury reports, market movements, team changes) and exit at zero cost.

Key property. The refund is dollar-for-dollar: every token refunded removes exactly its mint price from the pot, preserving the per-NFT backing ratio $B(t)/N_{\text{total}}(t)$ for uniform-priced games.

2.4 Prize Distribution — The Scorecard Formula

After the real-world event concludes and a scorecard is ratified, the game enters the COMPLETE phase.

The scorecard. A scorecard is a vector of weights $\mathbf{w} = (w_1, w_2, \dots, w_N)$ satisfying:

$$\sum_{i=1}^N w_i = W_{\text{total}} = 10^{18} \quad (8)$$

Each $w_i \in [0, W_{\text{total}}]$ represents the fraction of the prize pool allocated to tier i 's holders.

Per-token weight. The weight assigned to a single NFT in tier i is:

$$w_i^{\text{token}} = \frac{w_i}{\hat{n}_i} \quad (9)$$

where \hat{n}_i is the *effective* number of tokens eligible for redemption in tier i at the time the scorecard is ratified:

$$\hat{n}_i = n_i^{\text{minted}} - n_i^{\text{remaining}} - (n_i^{\text{burned}} - d_i) \quad (10)$$

Here n_i^{minted} is the initial supply, $n_i^{\text{remaining}}$ is the unminted supply, n_i^{burned} is the total burned count, and d_i is the number of tokens redeemed *in the complete phase specifically*. This formula ensures that as tokens are redeemed in the complete phase, the denominator adjusts to maintain fair distribution for remaining holders.

Cash-out value. When a player burns a set of token IDs $\{k_1, k_2, \dots, k_m\}$, the total claim is:

$$C(\{k_j\}) = \frac{\sum_{j=1}^m w_{i(k_j)}^{\text{token}}}{W_{\text{total}}} \cdot (B_{\text{prize}} + A_{\text{redeemed}}) \quad (11)$$

where $i(k_j)$ is the tier of token k_j , B_{prize} is the current treasury balance (post-fee), and A_{redeemed} is the cumulative amount already redeemed by prior players.

The term $(B_{\text{prize}} + A_{\text{redeemed}})$ reconstructs the *original* post-fee pot, ensuring that the order of redemptions does not affect the payout per token. This is a critical design property: it makes Defifa a *path-independent* mechanism.

Special cases:

- **Winner-take-all:** $w_j = W_{\text{total}}$ for a single tier j , all others zero.
- **Proportional split:** $w_i = W_{\text{total}} \cdot n_i / N_{\text{total}}$, weights by participation count.
- **No contest (by convention):** All w_i set proportionally to return mint prices, effectively implementing a full refund through the standard scorecard mechanism.

2.5 Fee Extraction Pipeline

Before prize distribution begins, the Deployer contract extracts protocol fees by calling `fulfill-CommitmentsOf`. This triggers a `sendPayoutsOf` call on the terminal, distributing the pot according to the scoring-phase splits.

Split structure. The splits configured at game launch allocate the pot as follows:

1. **Base protocol fee:** $1/\phi_{\text{base}}$ of the pot to the base protocol project (default: 5%)
2. **Defifa fee:** $1/\phi_{\text{defifa}}$ of the pot to the Defifa project (default: 5%)
3. **Custom splits (\mathcal{S}):** Any additional game-creator-defined splits
4. **Remainder:** Returned to the game's treasury via `addToBalanceOf`

Fee formulas. Let B_{pot} be the treasury balance at commitment fulfillment:

$$F_{\text{base}} = \frac{B_{\text{pot}}}{\phi_{\text{base}}} \quad (12)$$

$$F_{\text{defifa}} = \frac{B_{\text{pot}}}{\phi_{\text{defifa}}} \quad (13)$$

$$F_{\text{custom}} = \sum_{s \in \mathcal{S}} \frac{B_{\text{pot}} \cdot \text{percent}_s}{\text{SPLITS_TOTAL_PERCENT}} \quad (14)$$

The prize pool available for player claims is:

$$B_{\text{prize}} = B_{\text{pot}} - F_{\text{base}} - F_{\text{defifa}} - F_{\text{custom}} \quad (15)$$

With default parameters ($\phi_{\text{base}} = \phi_{\text{defifa}} = 20$, no custom splits):

$$B_{\text{prize}} = B_{\text{pot}} \cdot \left(1 - \frac{1}{20} - \frac{1}{20}\right) = 0.9 \cdot B_{\text{pot}} \quad (16)$$

Fee recycling. The fees paid to the Defifa and base protocol projects are processed as standard Juicebox payments, which mint project tokens (`$DEFIFA`, `$BASE_PROTOCOL`) to the beneficiary—in this case, the game's hook contract. These tokens are later distributed to players upon claim (Section 2.6).

2.6 Protocol Token Allocation

When fees are paid to the Defifa and base protocol projects, those projects mint their respective tokens to the game hook’s address. The hook contract accumulates these tokens and distributes them proportionally when players burn their NFTs in the COMPLETE phase.

Token allocation per player. For a player burning tokens with cumulative mint cost c :

$$X_{\text{defifa}} = \frac{c}{M} \cdot D_{\text{total}} \quad (17)$$

$$X_{\text{base}} = \frac{c}{M} \cdot P_{\text{total}} \quad (18)$$

where M is the total mint cost of all tokens ever minted, D_{total} is the total \$DEFIFA tokens held by the hook, and P_{total} is the total \$BASE_PROTOCOL tokens held by the hook.

Key property. Protocol token distribution is proportional to *original mint cost*, not to scorecard weight. This means that even holders of losing tiers ($w_i = 0$) receive protocol tokens when burning their NFTs, creating a partial consolation mechanism that rewards participation regardless of outcome.

3 Attestation Governance and Scorecard Ratification

3.1 Voting Power Model

The attestation mechanism uses a *per-tier proportional representation* model rather than a simple one-token-one-vote system.

Attestation units. Each tier i carries a maximum attestation power of:

$$V_{\text{max}} = 10^9 \quad (\text{MAX_ATTESTATION_POWER_TIER}) \quad (19)$$

This maximum is shared among all holders of tier i . A holder’s attestation weight for tier i is:

$$v_i^{\text{holder}} = V_{\text{max}} \cdot \frac{n_i^{\text{holder}}}{n_i^{\text{total}}} \quad (20)$$

where n_i^{holder} is the number of tier- i tokens delegated to (or held by) the attestor, and n_i^{total} is the total minted supply of tier i at the attestation snapshot timestamp.

Total attestation weight. A holder’s total attestation power across all tiers is:

$$v^{\text{holder}} = \sum_{i: n_i^{\text{holder}} > 0} V_{\text{max}} \cdot \frac{n_i^{\text{holder}}}{n_i^{\text{total}}} \quad (21)$$

Checkpoint-based snapshots. Attestation power is measured at a fixed historical timestamp (the scorecard’s `attestationsBegin` time), using historical checkpoints. This prevents vote-buying attacks where an actor acquires tokens immediately before voting.

Delegation. During the mint phase only, holders may delegate their attestation units to a chosen delegate address per tier. Delegation is per-tier, snapshot-locked, and mint-phase-only.

3.2 Quorum and Ratification Conditions

Quorum calculation. The quorum required for scorecard ratification is:

$$Q = \frac{N_{\text{minted_tiers}}}{2} \cdot V_{\text{max}} \quad (22)$$

where $N_{\text{minted_tiers}}$ is the number of tiers that have at least one minted token.

Example. For a game with 4 tiers (all minted):

$$Q = \frac{4}{2} \cdot 10^9 = 2 \times 10^9$$

This requires the equivalent of 2 full tiers’ worth of unanimous attestation—for instance, all holders of 2 tiers attesting, or 50% of holders across all 4 tiers.

Ratification conditions. A scorecard can be ratified when all three conditions are met:

1. The scorecard’s grace period has expired,
2. The attestation count meets or exceeds quorum,
3. No other scorecard has been ratified for this game.

3.3 Scorecard Lifecycle

Each submitted scorecard passes through five states:

State	Condition
PENDING	<code>attestationsBegin > block.timestamp</code>
ACTIVE	<code>attestationsBegin ≤ now ≤ gracePeriodEnds</code>
SUCCEEDED	Grace period expired AND attestations \geq quorum
DEFEATED	A different scorecard was ratified
RATIFIED	This scorecard was ratified

Table 2: Scorecard lifecycle states.

Multiple scorecards may coexist in ACTIVE or SUCCEEDED state simultaneously, but only one can ever be ratified. This creates a competitive dynamic where multiple proposed outcomes compete for attestation support.

3.4 Resistance to Strategic Manipulation

The attestation model incorporates several defenses:

Defense 1: Per-tier cap. No single tier’s holders can contribute more than V_{\max} attestation units, regardless of how many tokens they hold.

Defense 2: Checkpoint snapshots. Attestation power is computed at a fixed historical timestamp. Acquiring tokens after the snapshot provides zero additional voting power.

Defense 3: Mint-phase-only delegation. Delegation is locked after the mint phase, preventing last-minute delegation changes during the scoring phase.

Defense 4: 50% quorum across tiers. Requiring half of all minted tiers’ worth of attestation power means no coalition controlling fewer than half the minted tiers can unilaterally ratify a fraudulent scorecard.

Remaining attack surface. A coalition controlling sufficient attestation power across $\lceil N_{\text{minted}}/2 \rceil$ tiers can ratify an arbitrary scorecard. The critical insight is that attestation power within a tier is *proportional to token holdings*, not absolute. An attacker holding 100% of a tier’s supply—even just 1 token—receives the full $V_{\max} = 10^9$ attestation power for that tier.

Worst-case attack cost (heavily minted tiers). When all tiers are well-populated, the attacker must acquire majority holdings in at least $\lceil N/2 \rceil$ tiers:

$$C_{\text{attack}}^{\text{worst}} \geq \sum_{i \in \text{majority set}} \left\lceil \frac{n_i + 1}{2} \right\rceil \cdot p_i \quad (23)$$

Best-case attack cost (sparse tiers). When some tiers have zero or minimal mints, the attacker can buy 1 token in each unminted tier, becoming the sole holder and receiving full

attestation power:

$$C_{\text{attack}}^{\text{best}} = \sum_{i \in \text{cheapest } \lceil N/2 \rceil \text{ unminted}} p_i \quad (24)$$

This is potentially orders of magnitude cheaper. In a 32-tier game at 0.01 ETH where 16 tiers have zero mints, the attacker spends $16 \times 0.01 = 0.16$ ETH to meet quorum—regardless of pot size. **This is the most significant governance vulnerability identified** and is discussed further in Section 9.2.

For the attack to be profitable, the attacker must redirect more than C_{attack} in prize value:

$$B_{\text{prize}} > C_{\text{attack}} \cdot \frac{W_{\text{total}}}{\sum_{i \in \text{majority}} w_i^{\text{proposed}}} \quad (25)$$

For the sparse-tier attack, this condition is almost always satisfied when the pot is nontrivial. Games with broad, uniform participation across all tiers are resistant; games with uneven participation are vulnerable.

4 Price Dynamics and Value Flows

4.1 NFT Intrinsic Value During Minting

During the mint phase, the intrinsic value of a tier- i NFT depends on the holder’s subjective probability assessment.

Expected value at mint. Let π_i be a player’s subjective probability that tier i wins. The expected post-fee payout for one tier- i NFT in a winner-take-all game is:

$$\mathbb{E}[V_i] = \pi_i \cdot \frac{B_{\text{prize}}}{n_i} + X_i^{\text{protocol}} \quad (26)$$

A rational risk-neutral player mints tier i when $\mathbb{E}[V_i] > p_i$, which requires:

$$\pi_i > \frac{p_i - X_i^{\text{protocol}}}{B_{\text{prize}}/n_i} \quad (27)$$

This threshold probability decreases as the pot grows (more participants in other tiers create larger prizes) and increases as more tokens of tier i are minted (diluting the per-token payout within the tier).

4.2 Post-Scorecard Valuation

After the scorecard is ratified and fees are extracted, each NFT has a deterministic value:

$$V_i^{\text{token}} = \frac{w_i}{\hat{n}_i \cdot W_{\text{total}}} \cdot (B_{\text{prize}} + A_{\text{redeemed}}) + V_i^{\text{protocol}} \quad (28)$$

Winning tier (full weight). In a winner-take-all game with $w_j = W_{\text{total}}$:

$$V_j^{\text{token}} = \frac{B_{\text{prize}} + A_{\text{redeemed}}}{\hat{n}_j} + V_j^{\text{protocol}}$$

Losing tier (zero weight). When $w_i = 0$:

$$V_i^{\text{token}} = V_i^{\text{protocol}}$$

Losing-tier tokens have zero prize value but retain protocol-token value.

4.3 Secondary Market Implications

Pre-ratification. NFT value is driven by subjective outcome probabilities. Prices reflect the market’s consensus probability-weighted expected payout, analogous to prediction-market shares.

Post-ratification. NFT value is deterministic and publicly computable. Any secondary-market price deviating from the redemption value creates an arbitrage:

- If $P_{\text{market}} < V_i^{\text{token}}$: buy on the market, burn for profit.
- If $P_{\text{market}} > V_i^{\text{token}}$: never occurs rationally (burn dominates holding).

Post-ratification secondary markets should converge immediately to redemption value, eliminating any residual price discovery.

5 Rational Actor Analysis

5.1 Mint-Phase Strategy: Entry Timing

In a fixed-price game, there is no direct price advantage to minting early vs. late within the mint phase. However, strategic considerations arise.

Equilibrium. In a Nash equilibrium of the minting game with risk-neutral players, each player mints the tier maximizing their expected payoff. Denoting by π_i the true probability of tier i winning and by $f_i = n_i \cdot p_i / B$ the fraction of the pot allocated to tier i :

$$\mathbb{E}[\text{return}_i] = \frac{\pi_i}{f_i} \cdot (1 - \phi) - 1 \quad (29)$$

where $\phi = 1/\phi_{\text{defifa}} + 1/\phi_{\text{base}} + \phi_{\text{custom}}$ is the total fee rate.

In equilibrium, expected returns equalize across tiers: $\mathbb{E}[\text{return}_i] = \mathbb{E}[\text{return}_j]$ for all i, j with non-zero minting, which implies:

$$\frac{\pi_i}{f_i} = \frac{\pi_j}{f_j} \quad \Rightarrow \quad f_i = \pi_i \quad (30)$$

Result. In equilibrium, the fraction of the pot in each tier equals the market’s consensus probability of that tier winning. This is the classical parimutuel result: the pot allocation *reveals* the collective probability assessment.

5.2 Refund-Phase Strategy: Option Exercise

The refund phase creates a *free put option* on each minted NFT, struck at the mint price.

Option value. Let $V_i(t_{\text{refund_end}})$ be the expected value of a tier- i token at the end of the refund phase. The refund option has value:

$$O_i = \max(p_i - V_i(t_{\text{refund_end}}), 0) \quad (31)$$

A rational player exercises (refunds) when $V_i(t_{\text{refund_end}}) < p_i$, which occurs when new information shifts the expected outcome against their chosen tier.

The refund phase serves three purposes:

1. **Risk reduction:** allows players to participate speculatively with a guaranteed exit.
2. **Information revelation:** refund activity signals belief updates.
3. **Adverse selection mitigation:** partially solves the “winner’s curse” problem.

5.3 Scoring-Phase Strategy: Attestation Delegation

Equilibrium. In the unique subgame-perfect equilibrium of the attestation game (assuming common knowledge of the event outcome):

1. All holders attest to the *truthful* scorecard—the one reflecting the actual event outcome.
2. The truthful scorecard achieves quorum, as holders of winning tiers have the strongest incentive to attest.

5.4 Complete-Phase Strategy: Claim vs Hold

Dominant strategy. For risk-neutral players with positive time preference, burning immediately weakly dominates holding. The claim value does not depreciate (the path-independent formula ensures later claimants receive the same amount), but the time value of money favors immediate realization. Holding is justified only by expected protocol-token appreciation exceeding the discount rate:

$$\frac{dP_D}{dt} \cdot \frac{p_i}{M} \cdot D_{\text{total}} > r \cdot V_i^{\text{token}} \quad (32)$$

where r is the player's discount rate.

6 Solvency and Conservation Laws

6.1 The Conservation Guarantee

Theorem 1 (Prize Pool Conservation). *For any scorecard \mathbf{w} with $\sum_i w_i = W_{\text{total}}$ and any sequence of redemptions, the total amount paid out to all NFT holders equals B_{prize} .*

Proof. The total claim across all tokens is:

$$\sum_{i=1}^N n_i^{\text{eligible}} \cdot \frac{w_i}{\hat{n}_i \cdot W_{\text{total}}} \cdot (B_{\text{prize}} + A_{\text{redeemed}})$$

Since $n_i^{\text{eligible}} = \hat{n}_i$ at the start (before any complete-phase redemptions), and the term $(B_{\text{prize}} + A_{\text{redeemed}})$ is invariant, this equals:

$$\sum_{i=1}^N \frac{w_i}{W_{\text{total}}} \cdot B_{\text{prize}} = \frac{B_{\text{prize}}}{W_{\text{total}}} \sum_{i=1}^N w_i = B_{\text{prize}}$$

□

This guarantees that the treasury is exactly drained after all eligible tokens are redeemed—there is no residual and no shortfall.

6.2 Solvency Under Sequential Cash-Outs

Corollary 2 (Order Independence). *The payout to any individual NFT holder is independent of the order in which other holders redeem their tokens.*

Proof. The per-token claim formula (Eq. 11) uses $(B_{\text{prize}} + A_{\text{redeemed}})$ as the reference pot, which is constant regardless of how many tokens have been redeemed. As each token is redeemed, both n_i^{burned} and d_i increment by 1, leaving \hat{n}_i invariant. Therefore, each token receives the same payout regardless of when it is redeemed. □

6.3 Fee Impact on Total Claimable Value

The total value available to players (prize + protocol tokens) is:

$$V_{\text{total}} = B_{\text{prize}} + V_{\text{protocol}} = B_{\text{pot}} \cdot (1 - \phi) + V_{\text{protocol}} \quad (33)$$

With default fees ($\phi = 10\%$):

$$V_{\text{total}} = 0.9 \cdot B_{\text{pot}} + V_{\text{protocol}}$$

Whether the net present value exceeds the mint cost depends on whether $V_{\text{protocol}} > 0.1 \cdot B_{\text{pot}}$ —whether protocol token value compensates for the fee extraction.

7 Game-Theoretic Properties

7.1 Defifa as a Parimutuel Mechanism

Defifa implements a *generalized parimutuel mechanism* with several distinctive features:

Property	Traditional Parimutuel	Defifa
Outcome resolution	Centralized oracle	Decentralized attestation
Payout computation	House-computed odds	On-chain formula
Fee structure	Fixed takeout rate	Split-based, configurable
Asset type	Fungible bet tickets	Non-fungible ERC-721
Secondary market	Typically none	Full ERC-721 transferability
Refund option	Typically none	Configurable refund phase
Token rewards	None	Protocol token distribution

Table 3: Comparison: Traditional parimutuel vs. Defifa.

Parimutuel equivalence. Under uniform pricing ($p_i = p$), binary scorecard (one winner), and no refund phase, a Defifa game is equivalent to a classical parimutuel pool with odds:

$$\text{odds}_i = \frac{B_{\text{prize}}}{n_i \cdot p} = \frac{(1 - \phi) \cdot \sum_k n_k}{n_i} \quad (34)$$

7.2 Information Aggregation

The minting and refund dynamics create a multi-round price-discovery mechanism:

Round 1 (Mint phase). Players reveal information through tier selection. Under equilibrium, the pot distribution converges to the collective probability distribution.

Round 2 (Refund phase). Players who received new information can exit, and the refund pattern reveals belief updates.

Round 3 (Secondary market). If NFTs trade on secondary markets during the scoring phase, prices reflect the most current probability assessments.

This three-round structure is informationally richer than single-shot betting mechanisms.

7.3 Multi-Game Dynamics and Protocol Flywheel

Defifa generates a *protocol-level flywheel* through its fee-token mechanism:

1. Game fees \rightarrow minted to protocol projects as payments,
2. Protocol tokens are issued to the game hook,
3. Players claim protocol tokens upon burning NFTs,
4. Protocol token value reflects aggregate fee revenue across all games,

5. Higher token value \rightarrow higher expected returns \rightarrow more participation \rightarrow more fees.

Flywheel dynamics. Let G be the number of active games, \bar{B} the average pot size, and ϕ the fee rate. The aggregate fee revenue is:

$$R = G \cdot \bar{B} \cdot \phi \quad (35)$$

The fraction of the pot recovered through protocol tokens is:

$$\frac{V_{\text{protocol}}^{\text{game}}}{\bar{B}} = \phi^2 \cdot \mu \cdot G \quad (36)$$

where μ is the revenue multiple of protocol token valuation. This shows that the protocol-token recovery rate increases linearly with the number of games G , creating a positive network effect.

8 Parameter Design Space

8.1 Tier Count and Price Calibration

Tier count. The number of tiers N affects quorum difficulty ($Q \propto N$), per-tier dilution, and attack cost. Optimal regime: $4 \leq N \leq 32$ balances governance tractability with outcome granularity.

Price calibration. Uniform pricing ($p_i = p$) creates clean parimutuel dynamics where pot fractions equal minting fractions. Non-uniform pricing allows odds-adjustment at design time. Recommended: uniform pricing between 0.01 and 1 ETH per NFT.

8.2 Timing Parameters

Mint duration: Should approximate time until event, capped at ~ 30 days.

Refund duration: 1–7 days provides meaningful optionality without excessive uncertainty.

Attestation start time: 1–24 hours delay for preparation.

Attestation grace period: 1–7 days for broad participation.

8.3 Fee Calibration and Protocol Sustainability

The default fee structure is competitive with existing markets:

Platform	Takeout Rate
Horse racing (parimutuel)	15–25%
Sports betting (vig)	4–10%
Prediction markets (fees)	1–5%
Defifa (default)	10%

Table 4: Fee comparison across prediction platforms.

The effective fee rate, accounting for protocol token rebates, is:

$$\phi_{\text{eff}} = \phi \cdot (1 - \alpha) \quad (37)$$

where α is the fraction of fee value retained in protocol tokens. For $\alpha = 0.5$: $\phi_{\text{eff}} = 5\%$, competitive with low-fee prediction markets.

9 Open Problems and Mechanism Design Recommendations

The formal analysis in Sections 2–8 reveals several structural properties of the Defifa mechanism that merit attention. This section catalogs open problems discovered through systematic code review and game-theoretic analysis, ordered by severity, and proposes concrete protocol-level mitigations.

9.1 Governance Deadlock and Fund Recovery: A Deep Study

Severity: Significant (design consideration).

9.1.1 Historical Context

The original Defifa (Juicebox V3 era) included `NO_CONTEST` and `NO_CONTEST_INEVITABLE` phases. In V3, each game phase had to be manually advanced by calling `queueNextPhaseOf()`. If nobody called this function before a funding cycle “rolled over,” the `_noContestInevitable()` check detected the rollover and `_queueNoContest()` reconfigured the project for permanent full-price refunds (`duration=0`, `cashOutTaxRate=0`, `pausePay=true`). The V5 port pre-queues all rulesets at launch, eliminating the rollover risk—but also eliminating the sole trigger for no-contest. The dead code was removed as part of the V5 cleanup (see `AUDIT_FINDINGS L-D5`). This section formally analyzes whether a new form of no-contest should be reintroduced.

9.1.2 Exhaustive Deadlock Scenario Analysis

We identify five distinct scenarios in which game funds could become permanently inaccessible:

Scenario A: No scorecard submitted. The game reaches `SCORING`. Nobody calls `submitScorecardFor()`. All tier cash-out weights remain zero. The hook returns `cashOutCount = 0` and `afterCashOutRecordedWith` reverts with `NOTHING_TO_CLAIM`. Funds remain in the treasury indefinitely.

Scenario B: Quorum unreachable. A scorecard exists but attestation power is fragmented. No single scorecard accumulates 50% of eligible attestation weight. The governor’s `stateOf()` returns `ACTIVE` indefinitely—there is no expiry on attestation.

Scenario C: Dead attestation delegate. The `defaultAttestationDelegate` is set to an inaccessible address. Since delegation can only be changed during `MINT`, the accumulated attestation power is irrecoverably locked after `MINT` ends.

Scenario D: Attestation power in dead addresses. If $> 50\%$ of game pieces are transferred to contracts that cannot call `attestToScorecardFrom()`, exercisable attestation power drops below quorum permanently.

Scenario E: Split target reverts on ratification. `ratifyScorecardFrom()` calls `fulfillCommitmentsOf()`, which calls `sendPayoutsOf()`. If a split target reverts, the entire ratification transaction fails despite a governance-approved scorecard.

Scenario	Funds stuck?	Delegate resolves?	Automated?
A: No scorecard	Yes	Yes, if active	No
B: Quorum unreachable	Yes	Yes, if has power	No
C: Dead delegate	Yes	No	No
D: Dead attestation holders	Yes	No	No
E: Split target reverts	Yes	No	No

9.1.3 Effectiveness of the Default Attestation Delegate

When set, the `defaultAttestationDelegate` receives attestation units from every minter who does not specify a custom delegate. If no minter re-delegates, the delegate holds 100% of attestation power—easily exceeding quorum. It resolves Scenarios A and B in the common case.

However, it provides no hard guarantee because it depends on four assumptions: (1) the delegate is set (`address(0)` is valid), (2) the delegate remains operational, (3) the delegate acts honestly (it could self-ratify a malicious scorecard), and (4) minters do not re-delegate. The delegate is an excellent first line of defense—a trusted, social mechanism—but not a trustless, automated one.

9.1.4 Candidate Mechanism A: Minimum Participation Threshold

At game initialization, the organizer sets `minParticipation`—a minimum treasury balance required to proceed to SCORING. If the balance is below this threshold, the game enters a no-contest state where cash-outs return mint prices. Implementation: one new `uint256` in the ops data; `currentGamePhaseOf()` checks the balance before returning SCORING.

What it solves: Ghost games with negligible participation skip directly to refundability.

What it does not solve: Post-threshold governance deadlocks (Scenarios B–E). *Attack surface:* A majority holder can refund enough tokens to push the balance below threshold, unilaterally killing the game. *Mitigation:* Set the threshold conservatively low (~10% of expected pot).

9.1.5 Candidate Mechanism B: Scorecard Ratification Timeout

At game initialization, the organizer sets `scorecardTimeout`—a duration after SCORING begins. If no scorecard is ratified within this window, the game enters a no-contest state. Implementation: one new `uint256`; `currentGamePhaseOf()` checks `block.timestamp > scoringStart + timeout`.

What it solves: All five deadlock scenarios (A–E). This is the only mechanism providing a hard, trustless, time-bounded guarantee. *Interaction with the governor:* Partially-attested scorecards expire gracefully. The critical edge case is a SUCCEEDED scorecard that hasn't been ratified when the timeout fires—mitigated by generous timeouts (90–180 days) or a short “ratification grace period.” *Attack surface:* Minimal—an adversary cannot accelerate the timeout.

9.1.6 Do We Need a Formal State?

Both mechanisms can be implemented as *computed states* in `currentGamePhaseOf()`—no on-chain state transition required. The SCORING ruleset already has `cashOutTaxRate = 0` and `pausePay = true`, which are the correct parameters for refunds. Adding a named enum value provides clearer UI/indexer signaling, but the behavior itself requires no new handler logic beyond the existing MINT/REFUND refund path.

9.1.7 Assessment and Recommendation

The `defaultAttestationDelegate` resolves the vast majority of practical deadlocks and is sufficient for games with trusted organizers. However, for permissionless, trustless game creation, a hard guarantee is essential. We recommend:

1. **Scorecard ratification timeout** as the primary safety mechanism. Covers all five deadlock scenarios. Implementation cost: one `uint256`, one timestamp comparison. Recommended default: 90 days.

2. **Minimum participation threshold** as an optional complement. Provides early termination for non-viable games. Implementation cost: one `uint256`, one balance check.
3. Both mechanisms should be **optional** (default: 0 = disabled) to preserve backward compatibility.
4. The game should **remain fully playable without either mechanism**—they are safety nets, not requirements.

The combination of delegate (fast-path social resolution) + timeout (hard backstop) + threshold (early exit) provides defense in depth where each mechanism covers the failure modes of the others.

9.2 Cheap Cross-Tier Attestation Capture

Severity: Critical.

As identified in the corrected attack cost analysis (Equation 24), the per-tier attestation power cap creates an unintended vulnerability in games with uneven participation. The mechanism assigns equal maximum attestation power ($V_{\max} = 10^9$) to every tier *regardless of minted supply*. A tier with 1 token has the same governance weight as a tier with 10,000 tokens.

The attack. An adversary identifies $\lceil N/2 \rceil$ tiers with zero or minimal mints. They purchase 1 token in each, becoming the sole holder and receiving full V_{\max} per tier. Their total attestation power: $A_{\text{attacker}} = \lceil N/2 \rceil \cdot V_{\max} \geq Q$. They meet quorum unilaterally and ratify a scorecard directing W_{total} to their tokens.

Numerical example. A 32-tier game at 0.01 ETH. Popular tiers accumulate 1,000 tokens each; 16 tiers receive no organic mints. The attacker buys 1 token in each empty tier for 0.16 ETH total and ratifies a scorecard directing ~ 144 ETH to one of their tiers. **Return on investment:** $\sim 900\times$.

Root cause. The quorum function counts *any tier with nonzero supply* as eligible, conflating “meaningful community participation” with “a tier a single actor created a position in.”

Recommended fix. Introduce a minimum supply threshold for quorum eligibility:

$$Q = \frac{1}{2} \sum_{i=1}^N \mathbf{1}[n_i \geq n_{\min}] \cdot V_{\max} \quad (38)$$

where $n_{\min} \geq 2$. Alternatively, weight each tier’s attestation power by a concave function of supply, such as $\min(V_{\max}, \sqrt{n_i} \cdot V_{\max} / \sqrt{n_{\text{ref}}})$.

9.3 Prize Pool Under-Allocation

Severity: Significant.

The weight validation in `setTierCashOutWeightsTo` uses a strict greater-than check: `if (_cumulativeCashOutWeight > TOTAL_CASHOUT_WEIGHT) revert`. A scorecard with weights summing to *less than* W_{total} passes validation. The residual $B_{\text{prize}} \cdot (1 - \sum_i w_i / W_{\text{total}})$ remains permanently trapped in the treasury, breaking the conservation guarantee of Theorem 6.1 (which assumes $\sum_i w_i = W_{\text{total}}$).

Recommended fix. Change to exact equality: `if (_cumulativeCashOutWeight != TOTAL_CASHOUT_WEIGHT) revert`.

9.4 Attestation Timing Misconfiguration

Severity: Significant.

Both `attestationsBegin` and `gracePeriodEnds` are computed relative to `block.timestamp` at submission time—not relative to each other. If $\tau_{\text{grace}} < \tau_{\text{attest_start}}$, the grace period expires *before attestations begin*, creating a zero-length effective attestation window. Additionally, `initializeGame` performs no validation on the relationship between these parameters.

Recommended fix. Compute `gracePeriodEnds` relative to `attestationsBegin`: $t_{\text{grace_end}} = t_{\text{attest_begin}} + \tau_{\text{grace}}$, or validate in `initializeGame` that $\tau_{\text{grace}} \geq \tau_{\text{attest_start}}$.

9.5 Pre-Scoring Scorecard Submission

Severity: Moderate.

The `submitScorecardFor` function contains no game-phase check. Scorecards can be submitted and accumulate attestations during the MINT phase—before the underlying event has occurred. While `setTierCashOutWeightsTo` enforces SCORING for weight application, pre-accumulated attestations let a coordinated group achieve SUCCEEDED state before scoring opens, then ratify instantly.

Recommended fix. Add a phase check requiring SCORING phase for scorecard submission.

9.6 Fee Extraction Fragility

Severity: Moderate.

In `fulfillCommitmentsOf`, the function calls `sendPayoutsOf` with `minTokensPaidOut` set to the full treasury balance. The split structure returns $\sim 90\%$ back via `addToBalanceOf`. If the terminal interprets `minTokensPaidOut` as the minimum that permanently leaves the project, this transaction reverts—permanently blocking fee extraction and game completion.

Recommended fix. Set `minTokensPaidOut` to 0 or to the expected fee amount.

10 Conclusions and Practical Implications

This paper has formalized the cryptoeconomic mechanisms of Defifa: a prediction-game protocol that transforms NFT minting into a parimutuel wagering mechanism with governance-ratified outcomes.

Prize Distribution Mechanics. Defifa implements a path-independent, weight-proportional prize distribution through Equation 11. Using $(B_{\text{prize}} + A_{\text{redeemed}})$ as the reference pot ensures every token holder receives the same payout regardless of redemption order. Theorem 6.1 proves total payouts exactly exhaust the prize pool (provided $\sum_i w_i = W_{\text{total}}$; see Section 9.3).

Governance Security. The attestation model (Section 3) achieves a balance between decentralization and efficiency. The per-tier cap on attestation power ($V_{\text{max}} = 10^9$) prevents any single tier from dominating governance, while the 50% quorum across minted tiers ensures broad participation. However, Section 9 identifies a critical governance vulnerability: cheap cross-tier attestation capture (9.2), where an attacker buying 1 token in each of $N/2$ unpopular tiers can unilaterally meet quorum. The corrected attack cost (Eq. 24) shows that governance security depends not just on tier count and prices, but critically on participation uniformity across tiers. The deep study in Section 9.1 identifies five distinct deadlock scenarios and evaluates two candidate safety mechanisms (participation thresholds and ratification timeouts), concluding that both are valuable optional additions but that the existing system remains fully playable without them when games have active organizers and trusted delegates.

Market Efficiency. The equilibrium analysis demonstrates convergence to the classical parimutuel result: pot fractions equal consensus probabilities. The three-round information structure (mint \rightarrow refund \rightarrow secondary) provides richer information aggregation than single-shot mechanisms.

Protocol Sustainability. The fee-token flywheel creates positive network effects where more games increase protocol token value, which reduces effective fee rates, which attracts more players.

Practical Recommendations.

1. **Tier count:** 4–32 tiers for governance security and outcome expressiveness.
2. **Pricing:** Uniform pricing between 0.01–1 ETH for clean parimutuel dynamics.
3. **Refund phase:** 1–7 days for meaningful optionality.
4. **Attestation:** Trusted default delegate; 24-hour start delay; 3-day grace period. Ensure $\tau_{\text{grace}} \geq \tau_{\text{attest_start}}$ (Section 9.4).
5. **Fees:** Default 10% split is competitive; organizer splits should not exceed 5%.
6. **Participation:** Ensure all tiers attract meaningful participation to resist cheap governance capture (Section 9.2). Consider minimum-supply quorum thresholds.
7. **Deadlock protection:** For permissionless games, set a scorecard ratification timeout (90–180 days recommended) and optionally a minimum participation threshold. For trusted-organizer games, the `defaultAttestationDelegate` is sufficient (Section 9.1).

Synthesis. Defifa implements a rigorous approach to prediction gaming through the composition of three well-understood mechanisms: parimutuel pooling for price formation, attestation governance for outcome resolution, and Juicebox V5 for treasury management. The mathematical analysis confirms that the system conserves value and converges to informationally efficient equilibria. The protocol token layer adds a novel incentive dimension that aligns participant, organizer, and protocol interests around game volume growth.

The open problems identified in Section 9—particularly the cheap cross-tier attestation capture (9.2) and prize pool under-allocation (9.3)—represent the most important areas for protocol hardening before production deployment at scale. The recommended mitigations (minimum-supply quorum thresholds, exact weight validation) are backwards-compatible and address the identified vulnerabilities without altering the core mechanism design. The deep study of governance deadlock (9.1) confirms that the existing architecture is sound—the `defaultAttestationDelegate` resolves the majority of practical deadlocks—but that optional safety mechanisms (ratification timeout, participation threshold) provide valuable defense in depth for permissionless deployment without adding mandatory complexity.

The elegance of Defifa resides in its architectural composability: prediction games with arbitrary outcomes, arbitrary tier structures, and arbitrary payout distributions emerge from the same set of parameters, executed deterministically by immutable smart contracts with a single, time-bounded governance input. The game remains fully playable and efficient without additional states—the proposed safety mechanisms are optional parameters that expand the design space for risk-averse game creators while preserving the protocol’s minimalist architecture for those who prefer it.