Horse Racing Instant Game

Technical Whitepaper

A Verifiable On-Chain Pari-Mutuel Racing Protocol with Probabilistic Tracks, Progressive Jackpots, and Tokenized Incentives

horse.fun Research Team

July 2026

Version 1.0

Built on Solana

Abstract

This paper presents the formal specification of *Horse Racing Instant Game*, a high-frequency on-chain racing protocol built on Solana. The system integrates provably-fair verifiable randomness via Switchboard VRF, four distinct probability track configurations, dual progressive jackpots, a volume-linked reward token economy with burn-based deflation, and a fee-funded referral mechanism.

Each race lasts 2.5 minutes and uses a pari-mutuel pool model with three placement payouts (1st, 2nd, 3rd). The protocol operates continuously with 576 races per day, ensuring constant availability and sustainable economics.

This work defines the mathematical foundations, tokenomics, security assumptions, and economic sustainability of the protocol. All race outcomes, payouts, and token distributions are verifiable on-chain, ensuring complete transparency and trustless operation.

Contents

1 Executive Summary

Horse Racing Instant Game is a decentralized horse racing protocol operating on Solana with continuous 2.5-minute race cycles. The protocol eliminates trust requirements through on-chain execution and cryptographic verification while maintaining engaging gameplay and sustainable economics.

Key Mechanisms

- **Verifiable Randomness**: Switchboard VRF ensures provably fair race outcomes that cannot be predicted or manipulated
- **Four Track Types**: Track A (favorites), Track B (balanced), Track C (chaos), and Track D (underdog) offer varying risk profiles
- **Pari-Mutuel Pools**: Self-balancing pool system where players compete against each other, not the operator (95% return rate)
- **Dual Jackpots**: Normal (1/125 chance) and Super (1/12500 chance) jackpots accumulate from each race and distribute to winners
- **REWARD Token**: volume-linked emission (tokens scale with entry volume), distributed by entry share, with deflationary burns on jackpot wins

Economic Model

The protocol sustains itself through a 5% fee on total entry volume, allocated to:

- Up to 1% for referral rewards
- 4% for protocol operations, development, and treasury

This fee structure ensures sustainable protocol development and maintenance regardless of entry volume.

2 Nomenclature

Symbol	Description
P_t	Probability vector for track t (Track A, B, C, or D)
p_i	Winning probability for position i in probability vector
σ	VRF-generated permutation mapping horses to probabilities
seed_n	VRF seed for race n (256-bit random value)
B	Total entry volume per race (sum of all entries)
B_i	Total amount entered on horse i by all players
$b_{i,u}$	Amount entered by user u on horse i
b_u	Total amount entered by user u across all horses
F_{protocol}	Protocol fee (5% of gross entry volume B)
P_j	Prize pool for position $j \in \{1, 2, 3\}$
J_n	Normal jackpot pool (accumulated SOL)
J_s	Super jackpot pool (accumulated SOL)
$B_{\min}$	Minimum race volume for jackpot-trigger eligibility (volume gate)
E	REWARD token emission per race (volume-linked, see Section 9)
R_u	REWARD tokens of user u (jackpot formulas use the staked balance at entry close, Section 9.4)
β	Burn rate coefficient (0.10 for normal, 0.50 for super jackpot)
λ	Jackpot boost coefficient (staked-token multiplier, Section 8.4)
H	Shannon entropy (measure of probability distribution unpredictability)

Table 1: Mathematical notation and symbols used throughout this paper

3 Introduction

3.1 Motivation and Problem Statement

Traditional online horse racing platforms suffer from several critical limitations:

1. **Opacity**: Race outcomes and payout calculations are not verifiable
2. **Centralization**: Players must trust operators with funds and fairness
3. **Value Extraction**: Profits flow to centralized entities rather than participants
4. **Low Frequency**: Traditional racing schedules offer few events per day, with long waits between races

Horse Racing Instant Game addresses these challenges through a fully decentralized, high-frequency protocol built on Solana. The system provides cryptographic guarantees of fairness, transparent on-chain execution, and a player-aligned token economy.

3.2 Protocol Overview

Horse Racing Instant Game is a decentralized, fast-cycle racing protocol designed for trustless execution, transparent payouts, and a sustainable long-term economy. Each race includes 9 horses whose winning probabilities are determined by track presets and shuffled randomly using a VRF permutation.

The protocol introduces:

- Four thematic track probability distributions with varying risk profiles
- A 3-tier prize distribution based on pari-mutuel pooling
- A dual jackpot system with independent probabilistic triggering
- A volume-linked reward token with burn-based deflation
- A fee-funded referral mechanism aligned with growth incentives

The protocol operates natively on Solana due to its high throughput, low transaction costs, and deterministic execution model. With 576 races per day, the system achieves approximately 17,280 game rounds per month.

4 Race Cycle Overview

Each race consists of two phases:

- **Entry Phase** (120 seconds) Players submit entries on one or more horses.
- **Race Phase** (30 seconds) The outcome is fixed by the VRF seed the moment entries close; the race is then presented to players as an animation of that predetermined result, while the protocol settles payouts, jackpots, and REWARD distribution.

The system operates continuously:

$$\text{Races/day} \approx 576$$

4.1 Cross-Race Dependency

Races are linked through VRF seeds:

- Race n VRF seed determines race n outcomes (winner, jackpots)
- The same seed determines race $n + 1$ setup (track, probability permutation)

This ensures:

1. Players cannot predict the next track before entries close
2. Horse probability assignments remain unpredictable until race resolution
3. All randomness is verifiable on-chain through VRF proofs

5 Track Probability Model

5.1 Track Design Philosophy

The protocol features four distinct tracks, each representing a different competitive scenario. Each track defines how winning probabilities are distributed among the 9 horses in a race.

Formally, a track is a probability vector:

$$P_t = [p_1, p_2, \ldots, p_9], \quad \text{where } \sum_{i=1}^{9} p_i = 1$$

The probabilities are ordered from highest to lowest ($p_1 \geq p_2 \geq \cdots \geq p_9$), but before each race, a VRF-generated permutation σ randomly assigns these probabilities to the 9 horse positions, ensuring no predictable advantage.

After permutation, horse at visual position i has winning probability:

$$\Pr(\text{horse } i \text{ wins}) = p_{\sigma(i)}$$

5.2 Track Characteristics

Each track offers a different risk/reward profile for players:

Track A (Favorites-Heavy)

Probability vector:

$$[0.24, 0.18, 0.13, 0.11, 0.09, 0.08, 0.07, 0.05, 0.05]$$

Profile: Strong favorite with clear hierarchy. The top horse has nearly a 1-in-4 chance of winning.

Strategy: Safer entries on favorites yield moderate returns. Longshots offer high risk/high reward if entry volume concentrates on favorites.

Entropy: $H = -\sum p_i \log_2 p_i \approx 2.98$ bits (lowest entropy = most predictable)

Track B (Balanced)

Probability vector:

$$[0.18, 0.15, 0.13, 0.11, 0.10, 0.09, 0.08, 0.08, 0.08]$$

Profile: More balanced competition with a moderate favorite.

Strategy: Mid-tier horses offer good value. Less concentration on single favorites makes payout ratios more distributed.

Entropy: $H \approx 3.11$ bits (moderate entropy)

Track C (Chaos)

Probability vector:

$$[0.14, 0.13, 0.12, 0.11, 0.11, 0.11, 0.10, 0.09, 0.09]$$

Profile: Highly competitive field with no clear favorite.

Strategy: Any horse can win. Rewards skilled players who can identify value based on pool distribution rather than raw probabilities.

Entropy: $H \approx 3.16$ bits (highest entropy = most unpredictable)

Track D (Moderate Underdog)

Probability vector:

$$[0.17, 0.14, 0.12, 0.11, 0.11, 0.10, 0.09, 0.08, 0.08]$$

Profile: Slight favorite with strong mid-tier competition.

Strategy: Balanced between Track A and Track C. Good for players seeking moderate risk with potential for underdog upsets.

Entropy: $H \approx 3.13$ bits (moderate-high entropy)

5.3 Track Selection

Track selection is deterministic but unpredictable: the VRF seed from the **previous race** determines which track will be used for the current race.

Specifically:

$$\text{track\_index} = \text{seed}_{\text{previous}} \mod 4$$

This maps to one of the four tracks: Track A, Track B, Track C, or Track D.

This approach ensures:

- **Unpredictability**: Players cannot know the next track until the previous race resolves
- **Fairness**: All tracks have equal probability (25% each) over time
- **Verifiability**: Track selection can be verified on-chain from the previous VRF output

The variety of tracks maintains player engagement by offering different strategic environments and preventing monotony.

6 VRF Randomness Model

6.1 Switchboard VRF Integration

The protocol uses Switchboard VRF, a decentralized oracle network on Solana that provides verifiable randomness. Unlike centralized random number generators, Switchboard VRF ensures that:

1. Random values cannot be predicted before generation
2. Results can be cryptographically verified on-chain
3. No single party can manipulate the outcome

6.2 Randomness Request Process

At the end of each entry phase, the protocol:

1. Requests randomness from Switchboard VRF oracle
2. Receives a 256-bit random seed: $\text{seed}_{\text{current}} \in \{0,1\}^{256}$
3. Uses this seed to determine all race outcomes

The seed is committed on-chain with a cryptographic proof, making it publicly verifiable.

6.3 Dual-Purpose VRF Seed

Each race's VRF seed serves two purposes:

1. **Current race**: Determines winner, 2nd, 3rd place, and jackpot triggers
2. **Next race**: Determines track selection and horse probability permutation

This creates a dependency chain where race n determines the setup for race $n+1$, ensuring no advance knowledge of track or probability assignments.

6.4 Horse Permutation

For race n, we use seed_{n-1} (from the previous race) to shuffle the assignment between horses and probability values:

Step 1: Select track using $\text{track\_index} = \text{seed}_{n-1} \mod 4$

Step 2: Take the track probability vector $P_t = [p_1, p_2, \ldots, p_9]$

Step 3: Apply Fisher-Yates shuffle using seed_{n-1} to generate permutation σ

Step 4: Assign probabilities - horse at visual position i gets probability $p_{\sigma(i)}$

Example: If seed_{n-1} maps position 7 to index 1 on Track A, then horse #7 gets the highest probability (0.24), even though it appears in 7th position.

This prevents players from identifying "strong" positions or predicting the next track until the previous race completes.

6.5 Race Finishing Order

Once probabilities are assigned, the VRF seed determines the complete finishing order of the 9 horses through **sequential weighted sampling without replacement**, using only integer arithmetic to guarantee deterministic, on-chain verifiability.

Let the (already permuted) winning probabilities be expressed in basis points, $w_i \in \mathbb{Z}_{\geq 0}$ with $\sum_{i=0}^{8} w_i = 10{,}000$. Starting from the full set of horses $R = \{0, 1, \ldots, 8\}$, we fill finishing positions $k = 0, 1, \ldots, 8$ in order:

1. Derive 128 bits of randomness for position k from the VRF seed:

$$r_k = \text{SHA256}(\text{seed} \,\|\, \text{“ranking”} \,\|\, k)\,[0{:}16] \quad \text{(little-endian } u128\text{)}$$

2. Let $W = \sum_{j \in R} w_j$ be the total weight of the remaining horses, and set the target $t = r_k \bmod W$.
3. Walk the remaining horses, subtracting weights until $t < w_j$; that horse j is assigned finishing position k.
4. Remove j from R and repeat for the next position.

If $W = 0$ at some step (degenerate all-zero weights), the horse is chosen uniformly at random among those remaining. Finishing position 0 is the winner (1st place), position 1 is 2nd place, and position 2 is 3rd place; the same procedure yields the full ordering for positions 4 through 9.

Mathematical Foundation: This is an exact sampler for the **Plackett-Luce** distribution. The first horse is selected with probability proportional to its weight,

$$\Pr(i \text{ finishes 1st}) = \frac{w_i}{\sum_j w_j} = p_i,$$

and, conditioned on the horses already placed, each subsequent horse is selected proportionally to the remaining weights. The full finishing order $a = (a_0, a_1, \ldots, a_8)$ therefore has probability

$$\Pr(a) = \prod_{k=0}^{8} \frac{w_{a_k}}{\sum_{j \geq k} w_{a_j}},$$

and the marginal winning probability of each horse is exactly p_i.

Intuition: Horses with higher weight are more likely to be drawn early, making them more likely both to win and to place.

6.6 Jackpot Trigger Determination

To ensure independent jackpot triggers, we derive two independent values from the VRF seed using cryptographic hashing:

Normal Jackpot:

First, derive a seed for the normal jackpot:

$$\text{seed}_{\text{normal}} = \text{SHA256}(\text{seed}_{\text{current}} \,||\, \text{"normal\_jackpot"})$$

Then check the trigger condition:

$$\text{trigger}_n = (\text{seed}_{\text{normal}} \mod 125) = 0$$

Probability:

$$\Pr(\text{Normal jackpot}) = \frac{1}{125} = 0.008 = 0.8\%$$

Super Jackpot:

Similarly, derive a seed for the super jackpot:

$$\text{seed}_{\text{super}} = \text{SHA256}(\text{seed}_{\text{current}} \,||\, \text{"super\_jackpot"})$$

Then check the trigger condition:

$$\text{trigger}_s = (\text{seed}_{\text{super}} \mod 12500) = 0$$

Probability:

$$\Pr(\text{Super jackpot}) = \frac{1}{12500} = 0.00008 = 0.008\%$$

Since the two jackpots use independent hash derivations, they are statistically independent events. Both can trigger in the same race with probability:

$$\Pr(\text{Both}) = \frac{1}{125} \times \frac{1}{12500} = \frac{1}{1{,}562{,}500} \approx 0.000064\%$$

Expected time between triggers (at 576 races/day):

- Normal Jackpot: 125 races ≈ 5.2 hours
- Super Jackpot: 12500 races ≈ 21.7 days

A trigger results in a payout only if the race meets the volume-eligibility condition of Section 8.3; gated triggers are void, so realized payout intervals are correspondingly longer and the accumulated pots larger.

Note on uniformity: SHA256 is a cryptographic hash function with proven pseudo-random properties. Its output is uniformly distributed over $\{0,1\}^{256}$ under standard cryptographic assumptions. When applying modulo reduction to a uniform 256-bit value, the statistical bias is negligible: $\epsilon = n/2^{256} < 10^{-70}$ for $n \in \{125, 12500\}$, ensuring the stated probabilities hold with negligible deviation.

6.7 Liveness and Genesis

Oracle liveness. The protocol depends on the oracle fulfilling each randomness request. If no valid VRF proof arrives within a timeout T_{VRF} after the entry phase closes, the protocol **re-requests** randomness and the race remains pending — resolution is simply late, and the race schedule continues around it (see *chain continuity* below). The per-attempt timeout is deliberately long (minutes, not seconds): each new request draws a fresh seed, so rapid retries would let an adversary capable of briefly censoring fulfillment transactions suppress unfavorable outcomes and "re-roll" the race. A slow retry cadence makes such censorship impractically expensive.

Permanent oracle failure. Retries continue indefinitely — the protocol defines no in-contract cancellation path. In the extreme case of a permanent oracle failure (the provider discontinuing service entirely), resolution is restored through the protocol's standard upgrade path: the multi-signature upgrade authority already governing protocol upgrades migrates the program to an alternative VRF provider, and pending races then resolve normally, with fresh verifiable randomness from the new oracle. The trust assumption involved — an honest, live upgrade authority — is the same one the protocol already requires for upgrades in general; this contingency introduces no additional trust.

The protocol never substitutes a deterministic fallback (such as a blockhash or a fixed seed) for missing oracle output: a known fallback seed would make the emergency outcome computable in advance — converting a liveness failure into an integrity failure.

Chain continuity. A pending race consumes nothing: the following race derives its track selection and permutation from the **most recent fulfilled seed**, and the randomness chain resumes with its next fulfillment. Since track and permutation are public during the entry phase by design, reusing the latest fulfilled seed for setup leaks no outcome-relevant information — winners and jackpot triggers always come from the fresh seed drawn at each race's own entry close.

Genesis. The first race has no predecessor to inherit a seed from. At protocol initialization, a dedicated VRF request is made *before* race 1 opens for entries; its output serves as seed_0, so that even the first track selection and permutation are oracle-derived rather than operator-chosen.

7 Pari-Mutuel Pool Model

7.1 What is a Pari-Mutuel Pool?

Unlike traditional fixed-ratio models where an operator sets the payout ratios, pari-mutuel entry pools all entries together and divides the prize pool among winners proportionally to their entries.

This creates a self-balancing system: if many players back the same horse, the payout per winning entry decreases (similar to how large shared prize pools are split among multiple winners).

7.2 Pool Formation

During the entry phase, players submit entries on horses. Let:

- B_i = total amount entered on horse i (by all players)
- $B = \sum_{i=1}^{9} B_i$ = total entry volume for the race

Example: If 100 SOL is entered on horse #1, 200 SOL on horse #2, and so on, then B is the sum of all these amounts.

7.3 Prize Pool Distribution

The total entry volume B is distributed according to the following allocation:

$$
\begin{aligned}
P_1 &= 0.60 \times B && \text{(60\% to 1st place winners)} \\
P_2 &= 0.20 \times B && \text{(20\% to 2nd place winners)} \\
P_3 &= 0.10 \times B && \text{(10\% to 3rd place winners)} \\
J_n^{\text{increment}} &= 0.02 \times B && \text{(2\% to Normal Jackpot)} \\
J_s^{\text{increment}} &= 0.02 \times B && \text{(2\% to Super Jackpot)} \\
\text{Buyback} &= 0.01 \times B && \text{(1\% to REWARD buyback-and-burn)} \\
F_{\text{protocol}} &= 0.05 \times B && \text{(5\% protocol fee)}
\end{aligned}
$$

Total allocation: $60\% + 20\% + 10\% + 2\% + 2\% + 1\% + 5\% = 100\%$ of gross entry volume.

Example: If total entries $B = 10{,}000$ SOL:

- 1st place prize: $0.60 \times 10{,}000 = 6{,}000$ SOL
- 2nd place prize: $0.20 \times 10{,}000 = 2{,}000$ SOL
- 3rd place prize: $0.10 \times 10{,}000 = 1{,}000$ SOL
- Normal jackpot increment: $0.02 \times 10{,}000 = 200$ SOL
- Super jackpot increment: $0.02 \times 10{,}000 = 200$ SOL
- Buyback-and-burn: $0.01 \times 10{,}000 = 100$ SOL
- Protocol fee: $0.05 \times 10{,}000 = 500$ SOL

7.4 Individual Payout Calculation

If player u committed amount $b_{i,u}$ on horse i, and horse i finishes in position j (where $j \in \{1, 2, 3\}$):

$$\text{payout}_{u,j} = \frac{b_{i,u}}{B_i} \times P_j$$

Interpretation: The player receives a share of the position prize P_j proportional to their contribution to the total entries on that horse.

Example: Continuing from above with $B = 10{,}000$ SOL:

- First place prize: $P_1 = 0.60 \times 10{,}000 = 6{,}000$ SOL
- Suppose horse #3 wins and had total entries $B_3 = 800$ SOL
- Player Alice entered 100 SOL on horse #3
- Alice's payout: $\frac{100}{800} \times 6{,}000 = 750$ SOL
- Alice's profit: $750 - 100 = 650$ SOL

7.5 Unclaimed Prizes (Rollover Rule)

If no player backed the horse finishing in position j (i.e. $B_i = 0$ for that horse), the prize P_j has no eligible recipient. Rather than being retained by the protocol, any such prize is **routed to the buyback allocation**:

$$\text{Buyback}^{\text{new}} = \text{Buyback}^{\text{old}} + \sum_{j\,:\,\text{no eligible winner}} P_j$$

Unclaimed prizes therefore follow the buyback's regime (Section 9.6): before the liquidity pool exists they accumulate in the SOL reserve that seeds it; afterwards they fund additional buy-and-burn of the REWARD token. Since unclaimed prizes occur almost exclusively in thin, low-volume markets, this rule concentrates its effect exactly where it helps most: accelerating the bootstrap of protocol-owned liquidity.

The same principle applies protocol-wide: **any prize or reward share with no eligible recipient is routed to the buyback**. The one exception is a jackpot that triggers in a race where no player backed the winning horse: it is simply not paid out and continues to accumulate in its own pool.

This rule guarantees that the protocol never captures undistributed player funds: value that cannot be paid out immediately accrues to the player base — REWARD holders — through the volume-funded buyback.

7.6 Pari-Mutuel Properties

1. **No Operator Counterparty**: The protocol never takes a position against players — all prizes are paid from the entry pool itself, and the protocol earns only its transparent fee
2. **Self-Balancing**: Popular horses offer lower returns per unit entered
3. **No Payout-Ratio Manipulation**: Payouts determined by actual entry patterns
4. **Transparency**: All calculations verifiable on-chain

8 Jackpot Mechanics

8.1 Jackpot Accumulation

The protocol maintains two independent jackpot pools that grow with each race. From the gross entry volume B:

$$J_n^{\text{increment}} = 0.02 \times B \quad \text{(2\% goes to Normal Jackpot)}$$
$$J_s^{\text{increment}} = 0.02 \times B \quad \text{(2\% goes to Super Jackpot)}$$

After each race, the jackpot pools increase:

$$J_n^{\text{new}} = J_n^{\text{old}} + J_n^{\text{increment}}$$
$$J_s^{\text{new}} = J_s^{\text{old}} + J_s^{\text{increment}}$$

Example: If $B = 10{,}000$ SOL:

- Normal jackpot grows by: $0.02 \times 10{,}000 = 200$ SOL
- Super jackpot grows by: $0.02 \times 10{,}000 = 200$ SOL

8.2 Jackpot Triggers

As explained in Section 6, the VRF seed determines if a jackpot triggers:

- Normal Jackpot: $\Pr = 1/125$ per race
- Super Jackpot: $\Pr = 1/12500$ per race

The two jackpots trigger independently - both, one, or neither can trigger in the same race.

8.3 Trigger Eligibility: The Volume Gate

A jackpot trigger is **valid** only if the race that fires it carries meaningful entry volume:

$$\text{trigger pays} \iff \text{trigger fires} \ \wedge \ B \geq B_{\min}$$

If a trigger fires in a race below the threshold, nothing happens: the pot is untouched and the next race retains its independent trigger probability. Gated triggers are **void, not deferred** — deferring the payout to the next eligible race would create a publicly announced "jackpot race", distorting entry behavior. The gate applies identically to the Normal and Super jackpots.

Threshold definition:

$$B_{\min} = \max\left(B_{\text{floor}},\ k \cdot \text{median}\left(B_{\text{last 576 eligible races}}\right)\right), \qquad k = 0.5$$

where the median is taken over the most recent 576 races with $B \geq B_{\text{floor}}$ (approximately one full day of races), and B_{floor} is a small absolute minimum that covers the protocol's cold start, when the median itself is not yet meaningful.

Rationale. The trigger probability is volume-independent, so without a gate the accumulated pot — funded by the entire platform's volume — could be captured in a near-empty race by a trivial entry placed on every horse. The gate prices eligibility: making a race jackpot-eligible

requires routing $B_{\min}$ of real volume through the protocol's fee structure. For organic races this costs nothing (the volume is already there); for an adversary camping on empty races it is a recurring cost with negative expected return. Contributions from ineligible races still accrue to the pots, which only eligible races can win — low-volume and artificial races therefore subsidize the jackpots of organic ones.

Why a rolling median. A fixed threshold is wrong at every scale but one: too high during bootstrap (no race qualifies and the jackpot never pays), too low at maturity (dust races qualify and the gate is a placebo). A rolling *mean* can be pushed above every organic race by a single oversized wash race, freezing the jackpot for a day. The median is robust by construction: moving it requires controlling the *majority* of the 576-race window, and since an adversary can only ever *add* volume — at full fee cost, race after race — neither direction of manipulation is economically viable. The 24-hour window matches the daily volume cycle, so off-peak races are judged against the full day's median rather than against the quiet hours themselves.

Implementation. The median is maintained on-chain in $O(1)$ per race: a log-bucketed histogram over a 576-race ring buffer of eligible-race volumes, updated at settlement, with $B_{\min}$ cached in the race account at creation for public verifiability.

Side effect: during quiet periods jackpots pause and keep accumulating, so the advertised pot grows precisely when activity most needs stimulating.

8.4 Jackpot Distribution to Winners

When a jackpot triggers, it is distributed only among players who backed the **winning horse** (1st place). Each winner's share is proportional to their entry, **boosted linearly by their staked REWARD tokens**:

$$\text{score}_u = b_u \times (1 + \lambda R_u), \qquad \text{winner\_share}_u = J \times \frac{\text{score}_u}{\sum_{\text{winners}} \text{score}}$$

Where:

- J = jackpot amount being distributed
- b_u = amount winner u entered on the winning horse
- R_u = REWARD tokens staked by winner u, read at entry close (Section 9.4)
- λ = boost coefficient, a protocol parameter calibrated so that a typical staker earns an order of magnitude more per unit entered than a non-staker

Properties:

- **The entry is the ticket, tokens are the multiplier**: a winner with no staked tokens still receives their entry-proportional base share — the formula never pays zero to a committed winner. Conversely, a negligible entry yields a negligible score, so token holdings alone cannot capture the jackpot with a dust entry.
- **The boost is linear**: each additional staked token adds the same increment λb_u to the score — there is no point of diminishing returns. Staking more always increases both the jackpot share and the tokens placed at risk for the burn (Section 8.5), sustaining the stake–win–burn–rebuy cycle.

- For any meaningful stake ($\lambda R_u \gg 1$) the boost term dominates the base: serious stakers effectively compete on $b_u \cdot \lambda R_u$ alone, while the base share only protects casual winners from being shut out entirely.

Example (illustrative $\lambda = 1/200$): Normal jackpot of 10,000 SOL triggers; horse #5 wins, and only Alice and Bob backed it.

- Alice: entered 100 SOL, no staked tokens ⇒ score $= 100 \times (1 + 0) = 100$
- Bob: entered 400 SOL, 1,500 staked tokens ⇒ score $= 400 \times (1 + 7.5) = 3{,}400$

$$\text{Alice} = 10{,}000 \times \frac{100}{3{,}500} \approx 286 \text{ SOL}, \qquad \text{Bob} = 10{,}000 \times \frac{3{,}400}{3{,}500} \approx 9{,}714 \text{ SOL}$$

Bob's stake earns him an 8.5× boost per SOL entered; Alice, with no stake, still collects her base share — the entry is the ticket, the stake is the multiplier.

8.5 Token Burns (Deflationary Mechanism)

When a player receives jackpot winnings, a portion of their **staked** REWARD tokens — the same R_u frozen at entry close (Section 9.4) — is burned:

$$\begin{aligned} \beta_n &= 0.10 \quad (10\% \text{ burn for Normal Jackpot}) \\ \beta_s &= 0.50 \quad (50\% \text{ burn for Super Jackpot}) \end{aligned}$$

Burn amount for winner u:

$$\text{burn}_u = \beta \times R_u$$

where $\beta = \beta_n$ if only Normal Jackpot triggers, $\beta = \beta_s$ if only Super Jackpot triggers, or $\beta = \beta_n + \beta_s = 0.60$ if both jackpots trigger simultaneously.

Important: When both jackpots trigger in the same race, the burns are additive and calculated on the original token balance. A winner receiving both jackpots burns $10\% + 50\% = 60\%$ of their REWARD tokens.

Purpose: This creates deflationary pressure on the REWARD token supply, preventing unlimited accumulation and balancing emission.

Example 1 - Normal Jackpot only: If Alice wins the Normal Jackpot:

- Alice has 500 REWARD tokens staked
- Burn amount: $0.10 \times 500 = 50$ REWARD tokens
- Alice's new balance: $500 - 50 = 450$ REWARD tokens

Example 2 - Both Jackpots: If Alice wins both Normal and Super Jackpots in the same race:

- Alice has 500 REWARD tokens staked
- Burn amount: $(0.10 + 0.50) \times 500 = 0.60 \times 500 = 300$ REWARD tokens
- Alice's new balance: $500 - 300 = 200$ REWARD tokens

9 Reward Token Emission

9.1 Volume-Linked Emission

REWARD tokens are **not** emitted at a fixed rate. The emission of each race scales with that race's entry volume, capped to prevent any single oversized race from minting a disproportionate amount:

$$E = \min\left(c \times B,\ E_{\text{cap}}\right)$$

where B is the total entry volume of the race, c is the emission coefficient (REWARD tokens minted per SOL of volume), and E_{cap} is the per-race cap.

The coefficient c is **non-increasing**: it steps down at pre-announced **cumulative-volume milestones** (halving-style), so emission per SOL is richest in the protocol's earliest phase and tightens as it matures. Milestones depend only on total historical volume — never on the volume of the current race or of any trailing window — so the cost of minting a token is the same at every moment and cannot be improved by timing activity to quiet periods.

The cap E_{cap} bounds the emission of any single race, so that abnormally large (e.g. artificially inflated) races cannot mint a disproportionate amount of tokens; normal races emit in full. Above the cap, additional volume mints nothing while still funding the buyback (Section 9.6) — organic growth beyond the design scale therefore translates into net buy pressure rather than into supply.

The specific values of c, E_{cap}, and the milestone schedule are protocol parameters, documented separately.

Because emission tracks real volume, the token supply is **endogenous**: it grows only as fast as genuine activity, with no fixed daily or annual issuance.

9.2 Distribution to Players

The emission of each race is distributed to all participants in proportion to their entry volume:

$$R_u = E \times \frac{b_u}{B}$$

Where R_u is the REWARD tokens earned by player u, b_u is the amount they entered, and B is the total entry volume. **Below the cap**, this simplifies to $R_u = c \times b_u$: each player earns exactly c tokens per SOL committed.

Example ($c = 0.2$): a race with total volume $B = 20$ SOL mints $E = 0.2 \times 20 = 4$ tokens (below the cap).

- Alice enters 10 SOL $\Rightarrow 4 \times \frac{10}{20} = 2$ REWARD tokens
- Bob enters 5 SOL $\Rightarrow 4 \times \frac{5}{20} = 1$ REWARD token

9.3 Token Utility

REWARD tokens have a single purpose:

Jackpot Distribution: jackpot prizes are distributed in proportion to each winner's entry, boosted linearly by staked REWARD tokens — $b_u \times (1 + \lambda R_u)$, see Section 8.4. Staking more tokens increases a winner's share of the jackpot, with no point of diminishing returns.

This creates a retention mechanism where active players accumulate tokens that increase their share of future jackpots, rewarding long-term participation.

9.4 Staking

REWARD tokens must be **staked** to boost jackpot shares (Section 8.4). Staking is live and permissionless: tokens can be staked or unstaked at any time, with no lock-up period.

What counts for a given race is fixed the moment its entry phase closes:

$$R_u^{(n)} = \text{tokens staked by } u \text{ at the close of race } n\text{'s entry phase}$$

Once entries close, the entries are final: stake changes made after the close — in particular during the race phase, when the VRF outcome may already be observable on-chain — have no effect on race n and simply apply from race $n + 1$.

Two ordering guarantees make the snapshot airtight:

- **Boost and burn read the same value**: the frozen $R_u^{(n)}$ is used both for the jackpot share and for the burn on a win (Section 8.5).
- **Unstaking finalizes after settlement**: a withdrawal requested while race n is in flight executes only after race n settles, so a winner cannot observe a jackpot trigger on-chain and withdraw their stake ahead of the corresponding burn.

Every participant's boost and burn are therefore determined by their position at entry close, using exactly the information available to everyone.

9.5 Deflationary Balance

Tokens are burned during jackpot wins (Section 8.5). Over time:

- Emission: variable — scales with entry volume, capped per race
- Burns: variable, depending on jackpot frequency and winners' staked holdings

Both emission and burns scale with activity, moderating net supply growth over time.

9.6 Buyback and Burn

A fixed 1% of every race's entry volume — the buyback allocation — funds a **buyback-and-burn** of the REWARD token. This is the protocol's primary source of non-speculative token value: it turns game activity into continuous buy pressure, giving the token a value floor tied to volume rather than to sentiment alone.

The buyback operates in three phases:

1. **Seed** (before the pool exists): with no market to buy from, the buyback allocation accumulates as SOL. This reserve seeds the initial liquidity pool when the protocol opens its market — the protocol bootstraps its own liquidity from game activity, with no founder capital required.
2. **Deepen** (early market): for an initial period after the pool opens, the allocation delivers its buy pressure as **protocol-owned liquidity** — deposited into the pool rather than burned. Pool depth grows with volume, damping the price impact of every subsequent flow while the market is at its thinnest.
3. **Burn** (steady state): the allocation purchases REWARD from the pool and permanently burns it. Purchases execute continuously rather than in lumps, to avoid price spikes and front-running.

Effect on supply: both the buyback and the emission scale with volume, so the buyback offsets issuance by construction — the balance is set by the chosen rates, independent of the activity level. The buyback is the dependable, volume-funded supply sink; the jackpot burns (Section 8.5) are a secondary sink that depends on staking behavior.

10 Token Launch and Liquidity

The REWARD token has no allocation to the team or insiders. The only supply minted in advance is a **premint** reserved entirely for seeding the initial liquidity pool; every other token enters circulation through play, via the volume-linked emission.

Launch parameters:

- Premint: a fixed quantity of REWARD tokens, reserved entirely as the liquidity-pool seed
- Initial pool SOL: drawn entirely from the accumulated buyback reserve, not from external capital
- Resulting launch price: set mechanically by the seed-to-premint ratio, $p_{\text{launch}} = \dfrac{\text{pool SOL}}{\text{premint tokens}}$

Bootstrap to launch: until the pool exists, the buyback allocation accumulates as SOL rather than buying tokens. Once this reserve reaches the seed amount, the protocol pairs it with the premint to open the liquidity pool; the buyback then deepens the pool as protocol-owned liquidity for an initial period before switching to continuous buy-and-burn (Section 9.6). The token is therefore launched entirely from game activity — no founder capital and no insider supply.

Price discovery: the launch price is fixed mechanically by the seed-to-premint ratio above. From then on the market price is set by trading and by the volume-funded buyback; the protocol sets no price target.

11 Referral Incentive Mechanism

Referral rewards are funded from part of the developer fee, with no impact on pools, payouts, jackpots, or emissions.

For referred user u with volume b_u:

$$\text{reward}_r(u) = 0.01 \cdot b_u$$

The protocol retains:

$$4\%–5\% \text{ of } B$$

Properties:

- economically neutral,
- non-inflationary,
- sybil-resistant,
- scalable.

12 Economic Analysis

12.1 Expected Value for Players

The expected return for an entry depends on the horse's win probability and how entry volume is distributed among horses.

Simplified Case - Uniform Entries:

If all horses receive equal entry volume ($B_i = B/9$ for each horse), and we only consider 1st place prizes:

For an entry b on horse i with win probability p_i:

$$\mathbb{E}[\text{payout}_{1st}] = b \times p_i \times \frac{P_1}{B/9} = b \times p_i \times \frac{0.60 \times B}{B/9}$$

Simplifying:

$$\mathbb{E}[\text{payout}_{1st}] = b \times p_i \times 5.4$$

For Track A's favorite ($p = 0.24$):

$$\mathbb{E}[\text{payout}_{1st}] = b \times 0.24 \times 5.4 = 1.30b$$

Including 2nd and 3rd place would increase expected value further.

Reality: In practice, favorites attract disproportionate entries, reducing their actual payouts. The pari-mutuel system naturally balances toward value entries on underdogs with lower entry volume.

12.2 Protocol Margin Analysis

The protocol takes 5% of total volume as fee:

$$\text{Protocol fee} = \frac{F_{\text{protocol}}}{B} = 0.05 = 5\%$$

The remaining 95% flows back to players through three channels with different distribution profiles:

$$\underbrace{60\% + 20\% + 10\%}_{\text{direct prizes}} + \underbrace{2\% + 2\%}_{\text{jackpots}} + \underbrace{1\%}_{\text{buyback}} = 95\%$$

- **Direct prizes (90%)**: returned every race to players on the top-3 horses
- **Jackpots (4%)**: returned over time, only to jackpot winners, weighted by entry and boosted by staked REWARD (Section 8.4)
- **Buyback (1%)**: accrues to REWARD token holders via buy-and-burn

The expected net return therefore depends on how a player participates:

$$\mathbb{E}[\text{net return}] \approx \begin{cases} -0.10\,b & \text{direct prizes only (no staked REWARD, jackpot flows excluded)} \\ -0.05\,b & \text{aggregate, including jackpot and buyback flows} \end{cases}$$

The 5% aggregate edge is the true long-run figure across all players. The additional 5% gap faced by a non-staking, prizes-only player is exactly the value routed through jackpots and the token: staking REWARD and participating in jackpots is how a player closes it. This asymmetry is deliberate — it rewards sustained participants over one-shot players.

12.3 Jackpot Expected Value

Each race contributes to jackpots, which are eventually distributed back to players.

Because every accumulated jackpot is eventually paid out in full, the expected jackpot value returned per race equals the per-race contribution itself — independent of trigger frequency (rarity controls only how lumpy the payouts are, not the total):

$$\mathbb{E}[J_n \text{ payout}] = J_n^{\text{increment}} = 0.02 \times B, \qquad \mathbb{E}[J_s \text{ payout}] = J_s^{\text{increment}} = 0.02 \times B$$

Combined jackpot EV per race:

$$\mathbb{E}[J_{\text{total}}] = 0.02 \times B + 0.02 \times B = 0.04 \times B$$

This represents 4% of entry volume returned to players via jackpots per race — the full jackpot allocation.

12.4 Complete Payout Example

Race on Track A with $B = 10{,}000$ SOL total entry volume:

Distribution breakdown (all percentages from gross B):

- 1st place prize: $0.60 \times 10{,}000 = 6{,}000$ SOL
- 2nd place prize: $0.20 \times 10{,}000 = 2{,}000$ SOL
- 3rd place prize: $0.10 \times 10{,}000 = 1{,}000$ SOL
- Normal jackpot: $0.02 \times 10{,}000 = 200$ SOL
- Super jackpot: $0.02 \times 10{,}000 = 200$ SOL
- Buyback-and-burn: $0.01 \times 10{,}000 = 100$ SOL
- Protocol fee: $0.05 \times 10{,}000 = 500$ SOL

Total: $6{,}000 + 2{,}000 + 1{,}000 + 200 + 200 + 100 + 500 = 10{,}000$ SOL ✓

Player payout: If horse #3 wins with $B_3 = 800$ SOL entered on it, Alice who entered 100 SOL receives:

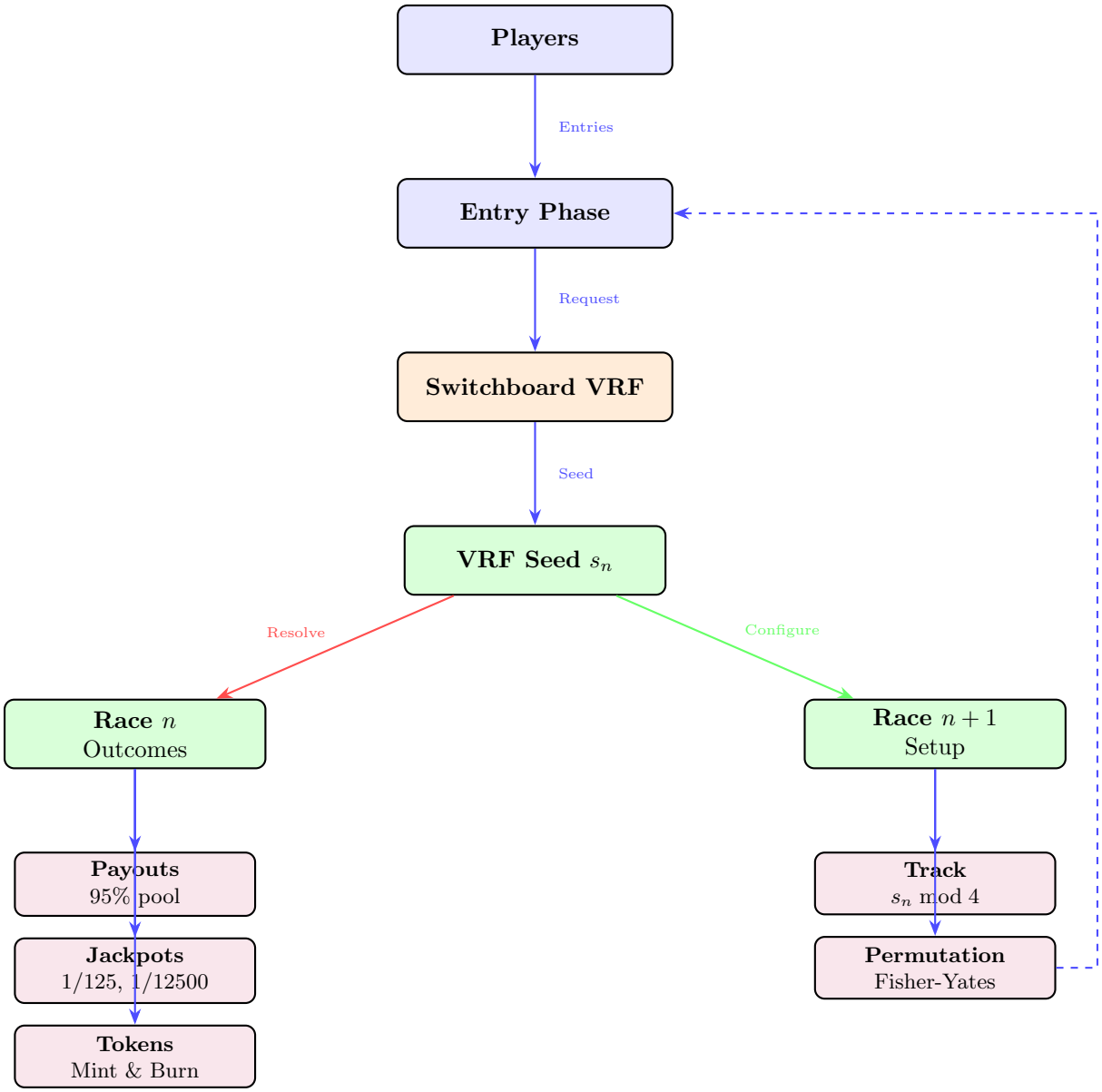
$$\text{payout} = \frac{100}{800} \times 6{,}000 = 750 \text{ SOL}$$

Alice's profit: $750 - 100 = 650$ SOL (650% ROI)

REWARD tokens: with emission coefficient $c = 0.2$, Alice earns $c \times b_{\text{Alice}} = 0.2 \times 100 = 20$ REWARD tokens (subject to the per-race cap).

13 System Architecture Diagram

The following diagram illustrates the complete protocol flow, including cross-race dependency and Switchboard VRF integration:



13.1 Architecture Components

1. Entry Phase

Players submit entries to the Solana smart contract. The contract:

- Collects all entries into pools per horse
- Deducts 5% developer fee (up to 1% for referrals)
- Locks entries until VRF resolution

2. Switchboard VRF Oracle

A decentralized oracle network that provides verifiable randomness:

- Multiple oracle nodes independently generate randomness
- Cryptographic proof ensures output cannot be predicted or manipulated
- Result committed on-chain with full verifiability

3. Dual-Purpose Seed

Each VRF seed s_n serves two functions:

- **Current race** n: Determines the full finishing order (Plackett-Luce sampling) — winner, 2nd/3rd place — and jackpot triggers
- **Next race** $n + 1$: Determines track selection ($s_n \bmod 4$) and probability permutation (Fisher-Yates)

4. Pari-Mutuel Distribution

95% of the entry pool is distributed proportionally among winners based on their entry shares.

5. Jackpot System

Two independent jackpots with probabilistic triggers:

- Normal Jackpot: 1/125 probability per race
- Super Jackpot: 1/12500 probability per race

Distribution: proportional to entry $\times$ $(1 + \lambda \cdot \text{staked REWARD})$.

6. Token Economics

- **Mint**: volume-linked ($c\times$ volume, capped per race) distributed to players by entry share
- **Burn**: 10% (normal) or 50% (super) of winner's REWARD tokens burned on jackpot wins

7. Cross-Race Dependency

The seed from race $n-1$ determines the configuration for race $n+1$, ensuring:

- No advance knowledge of track selection
- No predictable horse position advantages
- Continuous randomness chain throughout protocol operation

14 Security Considerations

14.1 Threat Model

We consider adversaries with the following capabilities:

1. **Financial adversary**: Can place arbitrary entries up to economic limits
2. **Network adversary**: Can observe transactions but not manipulate VRF
3. **Collusion**: Multiple players may coordinate strategies
4. **MEV searcher**: Can reorder transactions within a block

We do **not** consider:

- Compromise of Solana validators (beyond BFT assumptions)
- Cryptographic breaks of VRF primitives
- Protocol-level bugs (assume correct implementation)

14.2 Attack Vectors and Mitigations

1. Race Outcome Manipulation

Attack: Adversary attempts to predict or influence race outcomes.

Mitigation: VRF output is unpredictable before commit phase and deterministically verifiable after. The combination of race ID, timestamp, and blockhash ensures uniqueness and prevents pre-computation.

2. Pari-Mutuel Pool Manipulation

Attack: Large player places strategic entries to manipulate payout ratios.

Mitigation: Pari-mutuel structure is self-balancing. Any attempt to manipulate payout ratios by entering heavily on one horse reduces that horse's payout, making the attack unprofitable. The 5% fee creates negative EV for manipulators.

3. Jackpot Sniping

Attack: Adversary attempts to capture jackpot payouts with minimal stake — either by out-positioning other winners in an active race, or by placing dust entries on every horse in near-empty races so that any trigger landing there pays out to them alone.

Mitigation: The volume gate (Section 8.3) voids triggers in races below $B_{\min}$, turning the empty-race strategy into a recurring qualification cost with negative expected return. Within eligible races, jackpot distribution requires a committed entry on the winning horse (placed before VRF resolution), and the share scales with entry size, so token holdings alone cannot capture the jackpot with a negligible entry. Stake weights are frozen at entry close (Section 9.4), so observing a trigger on-chain during the race phase can change neither the boost nor the burn. The token burn on wins (Section 8.5) consumes the winner's staked tokens, rotating the advantage over time.

4. Front-Running and Informed Late Entries

Attack: MEV searchers reorder entries within a block; sophisticated players wait until the last moment to enter with full knowledge of the pool distribution.

Mitigation and design position: All entries are committed before the VRF is requested, and entry ordering does not affect outcomes or payouts, so classic MEV extraction has no surface. Late entries with full pool information are possible — and **deliberately so**: track probabilities and pool distributions are public to every participant by design, so the edge available to a late entrant is informational skill, not privileged access. This is the same dynamic as any pari-mutuel market, where payout ratios move until post time. The protocol prices this in transparently rather than claiming it away: value-seeking players sharpen the pools, and the protocol fee applies to them like everyone else.

5. Sybil Attacks on Referrals

Attack: User creates multiple identities to farm referral rewards.

Mitigation: Referral rewards are only 1% of volume and require actual game activity. The 5% protocol margin makes self-referral unprofitable. Economic cost exceeds reward.

6. Token Emission Exploitation

Attack: Adversary accumulates tokens to dominate jackpot distribution.

Mitigation: Jackpot burns (10% for normal, 50% for super) create deflationary pressure on large holders. Emission is volume-linked but capped per race, and minting tokens requires paying the protocol fee on the committed volume, making volume inflation to farm tokens self-limiting. The volume gate (Section 8.3) additionally prevents self-generated low-volume races from recapturing their own jackpot contributions, keeping the effective cost of volume inflation at its full fee rate.

14.3 Formal Security Properties

1. **Outcome Fairness**: the finishing order follows the Plackett-Luce distribution induced by the permuted weights, and is deterministically verifiable on-chain from the VRF seed. In particular, each horse wins with its assigned probability:

$$\Pr(\text{horse } i \text{ wins} \mid \sigma) = p_{\sigma(i)}$$

2. **Payout Correctness**: every unit of the prize allocation is accounted for — paid out to winners or routed to the buyback (Section 7.5):

$$\sum_u \text{payout}_u + \text{rollover} = 0.90 \cdot B + J_{\text{triggered}}$$

3. **Token Conservation**: total supply equals the premint plus all volume-linked emissions, minus the burns from both sinks:

$$S(t) = S_{\text{premint}} + \underbrace{\sum_r E_r}_{\text{emission}} - \underbrace{\sum \beta \cdot R_u}_{\text{jackpot burns}} - \underbrace{\text{burn}_{\text{bb}}(t)}_{\text{buyback burns}}$$

4. **Non-Negative Jackpots**:

$$J_n, J_s \geq 0 \text{ for all races}$$

14.4 Implementation Security

- All state transitions are atomic within Solana transactions
- Integer overflow protection via Rust's built-in checked arithmetic (panics on overflow by default)
- Access control on administrative functions using Anchor's account constraints
- Multi-signature wallet for protocol upgrades and treasury management
- Comprehensive testing including unit tests, integration tests, and fuzzing

15 Conclusion

This paper formalizes the complete design of Horse Racing Instant Game. The protocol combines verifiable randomness, sustainable tokenomics, and high-frequency on-chain execution to create a fair and engaging decentralized racing experience.

The key innovations presented include:

- Provably fair VRF-based race outcomes with verifiable probability distributions
- Self-balancing pari-mutuel pooling that resists manipulation
- Dual jackpot system with deflationary token burns
- Volume-linked emission tokenomics with natural equilibrium mechanisms
- Fee-neutral referral system enabling organic growth

The economic analysis demonstrates protocol sustainability under realistic volume assumptions, while the security analysis shows resistance to common attack vectors in decentralized gaming protocols.

15.1 Future Work

Several areas warrant further investigation:

1. **Empirical Validation**: Real-world data analysis of entry patterns, jackpot frequency, and token supply dynamics after protocol launch

2. **On-Chain Analytics**: Historical race statistics, track performance metrics, and player ROI analysis accessible directly from blockchain data
3. **Optimal Entry Strategies**: Game-theoretic analysis of equilibrium entry behavior under different track configurations and pool sizes
4. **Token Economics Modeling**: Long-term supply dynamics analysis considering varying jackpot frequencies and player retention patterns

16 References

1. Micali, S., Rabin, M., & Vadhan, S. (1999). *Verifiable random functions.* 40th Annual Symposium on Foundations of Computer Science.
2. Switchboard (2024). *Switchboard VRF: Verifiable Random Function on Solana.* https://switchboard.xyz
3. Yakovenko, A. (2018). *Solana: A new architecture for a high performance blockchain.* Solana Whitepaper.
4. Fisher, R. A., & Yates, F. (1948). *Statistical tables for biological, agricultural and medical research.* Oliver and Boyd, Edinburgh.
5. Hausch, D. B., Lo, V. S., & Ziemba, W. T. (1994). *Efficiency of racetrack betting markets.* Academic Press.
6. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., ... & Juels, A. (2020). *Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability.* IEEE Symposium on Security and Privacy.
7. Buterin, V. (2014). *A next-generation smart contract and decentralized application platform.* Ethereum Whitepaper.
8. Nakamoto, S. (2008). *Bitcoin: A peer-to-peer electronic cash system.* Bitcoin Whitepaper.
9. Bonneau, J., Clark, J., & Goldfeder, S. (2015). *On Bitcoin as a public randomness source.* IACR Cryptology ePrint Archive.
10. Catalini, C., & Gans, J. S. (2020). *Some simple economics of the blockchain.* Communications of the ACM, 63(7), 80-90.