IDL Protocol

A Decentralized Prediction Market for
Solana DeFi Metrics

Whitepaper v3.2

December 2025

Abstract

We present IDL Protocol, a decentralized prediction market platform built on Solana that enables users to bet on verifiable DeFi protocol metrics such as Total Value Locked (TVL), trading volume, and user counts. The protocol introduces a novel on-chain metrics oracle that derives all resolution data from pure Solana RPC calls without relying on third-party APIs or centralized data providers. Our system employs a commit-reveal betting scheme to prevent front-running, a parimutuel pool structure for fair payouts, and a dual-token economic model with deflationary mechanics. We provide rigorous mathematical formulations for metric calculation under Solana's RPC constraints, including stratified sampling algorithms for high-volume protocols and HyperLogLog-based probabilistic counting for unique user estimation. The protocol incorporates social trading features including prediction battles, guild-based pooled betting, and a referral system, all secured by oracle bonding, slashing mechanisms, and timelocked governance. This paper presents the complete technical specification, security analysis, and economic model of the IDL Protocol ecosystem.

OpenSVM Research
https://idlhub.com
https://github.com/openSVM/idlhub
rin@opensvm.com

Contents

1 Introduction **3**
1.1 Motivation and Vision . . . 3
1.2 Core Principles . . . 3
1.3 Contributions . . . 3

2 Related Work **4**
2.1 Prediction Markets . . . 4
2.2 On-Chain Oracles . . . 4
2.3 Vote-Escrow Tokenomics . . . 4

3 Problem Statement **4**
3.1 The IDL Fragmentation Problem . . . 4
3.2 Prediction Market Limitations . . . 5
3.3 Token Economic Failures . . . 5

4 Solution Overview **5**
4.1 IDLHub: The Registry . . . 5
4.2 Prediction Markets . . . 5
4.3 Social Trading Layer . . . 6

5 Token Economics **6**
5.1 Token Overview . . . 6
5.2 Supply Distribution . . . 6
5.3 Token Utility . . . 7
5.4 Fee Distribution . . . 7
5.5 Deflationary Mechanics . . . 7
5.6 Staking and VIP Tiers . . . 7

6 Core Protocol Architecture **8**
6.1 System Architecture . . . 8
6.2 Smart Contract Structure . . . 8
6.3 State Accounts . . . 8

7 Prediction Market Mechanism **9**
7.1 Market Lifecycle . . . 9
7.2 Commit-Reveal Scheme . . . 9
7.3 Parimutuel Pool Structure . . . 9
7.4 Oracle System . . . 10
7.5 Dynamic Odds . . . 10

8 Advanced Trading Features **10**
8.1 Limit Orders . . . 10
8.2 Stop Loss . . . 10
8.3 Partial Cashout . . . 11
8.4 Conviction Betting . . . 11

9 Social Trading Layer 11
9.1 Prediction Battles . . . 11
9.2 Guild System . . . 11
9.3 Referral System . . . 12
9.4 Leaderboards and Seasons . . . 12

10 StableSwap AMM 12
10.1 Purpose . . . 12
10.2 Curve StableSwap Invariant . . . 12
10.3 LP Token Economics . . . 12

11 On-Chain Metrics Oracle 13
11.1 Design Constraints . . . 13
11.2 TVL Calculation . . . 13
11.3 Volume Calculation: Stratified Sampling . . . 13
11.4 Complexity Analysis . . . 14
11.5 User Count: HyperLogLog . . . 14
11.6 Price Discovery . . . 15
11.7 Snapshot Consistency . . . 15
11.8 Confidence Scoring . . . 15

12 Security Analysis 15
12.1 Smart Contract Security . . . 15
12.2 Timing Constants . . . 16
12.3 Attack Vector Analysis . . . 16
12.4 Flash Loan Price Manipulation . . . 16
12.5 Sandwich Attack Prevention . . . 16

13 Governance 17
13.1 Vote-Escrow Mechanism . . . 17
13.2 Proposal Lifecycle . . . 17
13.3 Governable Parameters . . . 17

14 Limitations and Future Work 17
14.1 Current Limitations . . . 17
14.2 Future Work . . . 18

15 Conclusion 18

A Contract Addresses 19

B Glossary 19

1 Introduction

1.1 Motivation and Vision

The Solana ecosystem has experienced remarkable growth, hosting hundreds of decentralized applications spanning decentralized exchanges (DEXs), lending protocols, liquid staking derivatives, and NFT marketplaces. However, two critical infrastructure gaps remain:

(i) **Fragmented Interface Discovery:** Solana programs expose their interfaces through Interface Definition Language (IDL) files, yet these are scattered across GitHub repositories, often outdated, and inaccessible to automated agents.

(ii) **Prediction Market Vacuum:** While prediction markets have proven their utility for information aggregation and price discovery, no Solana-native solution exists for predicting DeFi-specific metrics.

IDL Protocol addresses both gaps by combining a comprehensive IDL registry (IDLHub) with a prediction market system that allows users to bet on quantifiable DeFi metrics using only on-chain data sources.

1.2 Core Principles

The protocol adheres to five foundational principles:

1. **Free Access:** The IDLHub registry remains freely accessible without token gating or paywalls.
2. **Real Yield:** Staking rewards derive from actual protocol revenue, not inflationary emissions.
3. **Deflationary Mechanics:** A portion of all protocol fees is permanently burned, reducing supply over time.
4. **Fair Launch:** No venture capital allocation, no presale—95% of tokens distributed to the public.
5. **Community Governance:** Protocol parameters are controlled by vote-escrowed token (veIDL) holders.

1.3 Contributions

This paper makes the following contributions:

- A complete specification of a prediction market protocol optimized for DeFi metric resolution on Solana.
- Novel algorithms for computing TVL, volume, and user counts using only pure Solana RPC methods, with formal error bounds.
- A commit-reveal betting scheme with parimutuel payouts and dynamic odds adjustment.
- Social trading primitives including 1v1 battles, guild-based pooled betting, and referral systems.
- Security analysis covering oracle manipulation, front-running, flash loan attacks, and economic attack vectors.

2 Related Work

2.1 Prediction Markets

Prediction markets have a rich history dating to the Iowa Electronic Markets [7]. Modern blockchain-based implementations include:

Augur [8] pioneered decentralized prediction markets on Ethereum using a dispute resolution mechanism with REP token staking. However, high gas costs and slow finality limit usability for frequent trading.

Polymarket operates as a hybrid system with off-chain order matching and on-chain settlement on Polygon. While achieving better UX, it relies on centralized market creation and resolution through UMA's optimistic oracle.

Gnosis introduced the LMSR (Logarithmic Market Scoring Rule) automated market maker [9], providing bounded loss for market makers but requiring subsidization.

Drift Protocol on Solana offers perpetual futures but focuses on price prediction rather than arbitrary DeFi metrics.

Our work differs by: (i) focusing specifically on DeFi protocol metrics, (ii) using pure on-chain data for resolution without external oracles, and (iii) leveraging Solana's high throughput for sub-second betting.

2.2 On-Chain Oracles

Oracle systems fall into two categories:

External Data Oracles (Chainlink, Pyth, Switchboard) aggregate off-chain data and post it on-chain. While robust for price feeds, they introduce trust assumptions and cannot provide historical DeFi metrics.

On-Chain Computation derives data directly from blockchain state. DeFiLlama computes TVL off-chain by indexing; we perform equivalent computation using only RPC calls, enabling trustless verification.

2.3 Vote-Escrow Tokenomics

The vote-escrow model originated with Curve's veCRV [3] and was extended by Solidly's ve(3,3) [5]. We adopt linear decay voting power but add prediction market utility beyond pure governance.

3 Problem Statement

3.1 The IDL Fragmentation Problem

Solana developers and AI agents seeking to integrate with existing protocols face a fragmented landscape:

Table 1: IDL Ecosystem Challenges

Problem	**Impact**
IDLs scattered across repositories	Hours of manual searching
Many IDLs outdated or incomplete	Integration failures
No standardized registry	Each team builds from scratch
AI agents cannot discover interfaces	Limits automation potential

3.2 Prediction Market Limitations

Existing prediction market platforms exhibit significant limitations for DeFi users:

Table 2: Existing Prediction Market Limitations

Platform	Limitation
Polymarket	Ethereum-based, high fees, no DeFi focus
Drift Markets	Limited to specific asset predictions
Custom solutions	Fragmented liquidity, poor UX

3.3 Token Economic Failures

Most DeFi governance tokens suffer from:

- **Inflationary Emissions:** Continuous token minting dilutes holder value.
- **Governance-Only Utility:** Tokens grant voting rights but generate no yield.
- **Lack of Engagement:** Beyond speculation, users have no reason to hold tokens.

4 Solution Overview

4.1 IDLHub: The Registry

IDLHub serves as the canonical registry for Solana program interfaces, hosting over 100 protocol IDLs including major platforms such as Jupiter, Marinade, Drift, Jito, Raydium, Orca, and Tensor.

Access is provided through multiple interfaces:

- **Web Interface:** Human-readable browser at `https://idlhub.com`
- **REST API:** Programmatic access via `/api/idl/{program}`
- **MCP Server:** Model Context Protocol for AI agent integration
- **JSON-RPC:** Standard RPC endpoint at `/api/mcp`

4.2 Prediction Markets

The protocol enables betting on verifiable DeFi metrics:

Definition 4.1 (Prediction Market)**.** A prediction market M is defined by the tuple:

$$M = (P, \mu, \theta, t_{\text{start}}, t_{\text{end}}, t_{\text{resolve}}) \tag{1}$$

where P is the target protocol, $\mu \in \{\text{TVL}, \text{Volume}, \text{Users}\}$ is the metric type, θ is the threshold value, and $t_{\text{start}}, t_{\text{end}}, t_{\text{resolve}}$ define the betting window and resolution time.

4.3 Social Trading Layer

The protocol incorporates social features to enhance engagement:

- **1v1 Battles:** Direct challenges between users on opposite sides of a prediction.
- **Guilds:** Pooled betting groups with profit sharing.
- **Leaderboards:** Rankings based on prediction accuracy.
- **Referrals:** Perpetual 5% fee share from referred users.
- **Seasons:** Time-limited competitions with prize pools.

5 Token Economics

5.1 Token Overview

The IDL token exists in two forms on Solana:

Table 3: Token Specifications

Parameter	**Value**
Network	Solana
Standard	SPL Token
Decimals	9
PUMP-IDL Address	`4GihJrYJGQ9pjqDySTjd57y1h3nNkEZNbzJxCbispump`
BAGS-IDL Address	`8zdhHxthCFoigAGw4QRxWfXUWLY1KkMZ1r7CTcmiBAGS`
Total Supply	2,000,000,000 IDL
Circulating Supply	∼1,950,000,000 IDL (97.5%)
Team Allocation	∼50,000,000 IDL (2.5%)

5.2 Supply Distribution

$$\text{Total Supply} = \underbrace{1\text{B (PUMP)}}_{\text{Active}} + \underbrace{1\text{B (BAGS)}}_{\text{Legacy}} \tag{2}$$

The distribution follows a fair launch model:

$$\text{PUMP-IDL:} \quad 800\text{M (Bonding Curve)} + 200\text{M (Raydium Migration)} \tag{3}$$

$$\text{BAGS-IDL:} \quad 950\text{M (Public)} + 50\text{M (Team)} \tag{4}$$

5.3 Token Utility

Table 4: Token Utility Matrix

Utility	Description	Requirement
Stake	Earn 50% of protocol fees	Hold IDL
Lock (veIDL)	Governance voting power	Lock staked IDL
Bet	Participate in predictions	Hold IDL
Battle	1v1 prediction challenges	Hold IDL
Guild	Create pooled betting groups	10 IDL fee
Lootbox	Random reward system	1–100 IDL per box
VIP Tiers	Fee discounts	Stake thresholds

5.4 Fee Distribution

Protocol fees follow a deterministic distribution:

Definition 5.1 (Fee Distribution)**.** For a winning claim of value W, the fee $F = W \times \beta$ where $\beta = 0.03$ (3%), distributed as:

$$F_{\text{stakers}} = 0.50 \times F \tag{5}$$

$$F_{\text{creator}} = 0.25 \times F \tag{6}$$

$$F_{\text{treasury}} = 0.15 \times F \tag{7}$$

$$F_{\text{burn}} = 0.10 \times F \tag{8}$$

$$F_{\text{referrer}} = 0.05 \times F_{\text{stakers}} \quad \text{(if applicable)} \tag{9}$$

5.5 Deflationary Mechanics

Multiple burn mechanisms reduce circulating supply:

1. **Prediction Market Fees:** 10% of all fees permanently burned.
2. **Lootbox Purchases:** 50% of purchase price burned.
3. **Guild Creation:** Portion of fee burned.
4. **Failed Stop Loss:** Small penalty burned.

Proposition 5.1 (Long-term Deflation)**.** *Given trading volume V and fee rate β, the annual burn rate is:*

$$B_{annual} = V \times 365 \times \beta \times 0.10 \tag{10}$$

5.6 Staking and VIP Tiers

Table 5: VIP Tier Benefits

Tier	Stake	Fee Discount	Bet Bonus	Perks
Bronze	100 IDL	0.5%	5%	Early access
Silver	1,000 IDL	1.0%	10%	+ Exclusive markets
Gold	10,000 IDL	1.5%	25%	+ Priority support
Platinum	100,000 IDL	2.0%	50%	+ Whale chat

The staking APY is calculated as:

$$\text{APY} = \frac{V_{\text{daily}} \times 365 \times \beta \times 0.50}{S_{\text{total}}} \times 100\% \tag{11}$$

where V_{daily} is daily volume and S_{total} is total staked value.

6 Core Protocol Architecture

6.1 System Architecture

Figure 1 illustrates the overall system architecture.

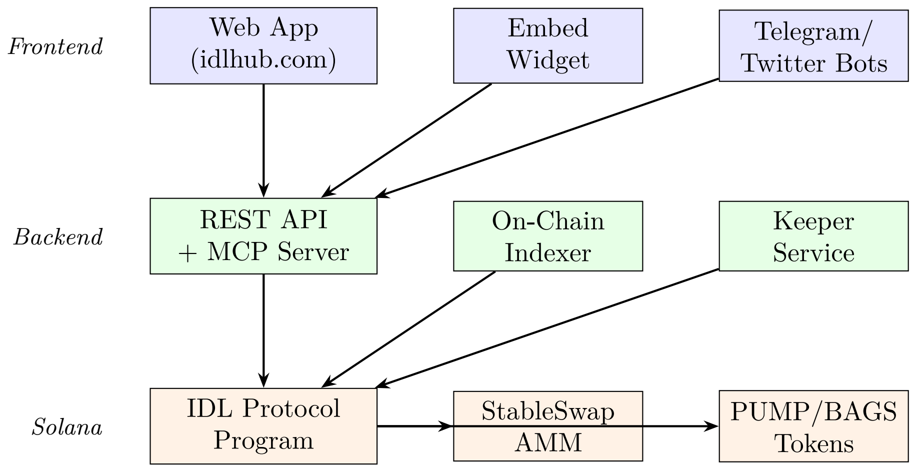

Figure 1: IDL Protocol System Architecture

6.2 Smart Contract Structure

The protocol consists of two primary Solana programs:

1. **IDL Protocol** (`BSn7neicVV2kEzgaZmd6tZEBm4tdgzBRyELov65Lq7dt`): Core staking, betting, and social features.
2. **IDL StableSwap** (`EFsgmpbKifyA75ZY5NPHQxrtuAHHB6sYnoGkLi6xoTte`): Curve-style AMM for token swaps.

6.3 State Accounts

Definition 6.1 (Protocol State)**.** The global protocol state is maintained in a Program Derived Address (PDA):

$$\text{ProtocolState} = \{\text{authority}, \text{treasury}, \text{vault}, S_{\text{total}}, V_{\text{total}}, R_{\text{pool}}, F_{\text{collected}}, B_{\text{total}}, \text{paused}\} \tag{12}$$

where S_{total} is total staked, V_{total} is veIDL supply, R_{pool} is pending rewards, $F_{\text{collected}}$ is lifetime fees, and B_{total} is lifetime burns.

Definition 6.2 (Staker Account)**.** Each staker maintains an account:

$$\text{StakerAccount} = \{\text{owner}, s, r_{\text{paid}}, r_{\text{pending}}, t_{\text{last}}\} \tag{13}$$

where s is staked amount, r_{paid} is reward-per-token checkpoint, r_{pending} is claimable rewards, and t_{last} is last stake timestamp.

Definition 6.3 (Vote-Escrow Position)**.** Locked staking positions are tracked as:

$$\text{VePosition} = \{\text{owner}, s_{\text{locked}}, v_0, t_{\text{start}}, t_{\text{end}}, \Delta t\} \tag{14}$$

where v_0 is initial veIDL amount, and $\Delta t = t_{\text{end}} - t_{\text{start}}$ is lock duration.

7 Prediction Market Mechanism

7.1 Market Lifecycle

A prediction market progresses through four phases:

1. **Creation:** Market creator specifies protocol, metric, threshold, and resolution time.
2. **Betting:** Users commit and reveal bets during the open window.
3. **Resolution:** Oracle commits and reveals the metric value at resolution time.
4. **Settlement:** Winners claim payouts; losers' stakes distributed.

7.2 Commit-Reveal Scheme

To prevent front-running, all bets use a two-phase commit-reveal mechanism:

Definition 7.1 (Bet Commitment)**.** A bet commitment is computed as:

$$c = H(\text{amount}\|\text{side}\|\text{nonce}\|\text{salt}) \tag{15}$$

where H is SHA-256, $\|$ denotes concatenation, and salt provides entropy.

Property 7.1 (Front-Running Resistance)**.** Given commitment c, an adversary cannot determine the bet parameters $(amount, side)$ without knowledge of $(nonce, salt)$, which remain secret until reveal.

The betting flow proceeds as:

1. **Commit Phase:** User submits c to create BetCommitment account. No tokens transferred.
2. **Delay:** Minimum 5-minute wait before reveal.
3. **Reveal Phase:** User submits $(amount, side, nonce, salt)$. Contract verifies $H(\cdot) = c$, transfers tokens to pool.

7.3 Parimutuel Pool Structure

Definition 7.2 (Parimutuel Payout)**.** For a market with YES pool P_Y and NO pool P_N, a winning YES bettor with stake s receives:

$$\text{Payout} = s + \frac{s}{P_Y} \times P_N \times (1 - \beta) \tag{16}$$

where β is the fee rate.

Theorem 7.1 (Zero-Sum Property)**.** *The parimutuel system is zero-sum: total payouts equal total stakes minus fees.*

$$\sum_{winners} Payout_i = (P_Y + P_N) \times (1 - \beta) \tag{17}$$

Proof. Without loss of generality, assume YES wins. Each YES bettor i with stake s_i receives:

$$\text{Payout}_i = s_i + \frac{s_i}{P_Y} \times P_N \times (1 - \beta) \tag{18}$$

Summing over all YES bettors:

$$\begin{aligned}
\sum_i \text{Payout}_i &= \sum_i s_i + \sum_i \frac{s_i}{P_Y} \times P_N \times (1 - \beta) && (19)\\
&= P_Y + \frac{P_N(1-\beta)}{P_Y} \sum_i s_i && (20)\\
&= P_Y + P_N(1-\beta) && (21)\\
&= P_Y + P_N - \beta P_N && (22)
\end{aligned}$$

The fee collected is βP_N, confirming zero-sum after fees. □

7.4 Oracle System

Resolution requires bonded oracles:

Definition 7.3 (Oracle Bond)**.** Before resolving a market, an oracle must deposit bond $B = 10$ IDL. The bond is:

- **Released** if resolution is not disputed.
- **Slashed** (50%) if resolution is successfully disputed.

The resolution follows a commit-reveal pattern identical to betting, with a 1-hour dispute window after reveal.

7.5 Dynamic Odds

Market odds update with each bet:

$$\pi_Y^{(t+1)} = \min\left(\pi_Y^{(t)} + 0.05, \frac{P_Y + \delta}{P_Y + P_N + \delta}\right) \tag{23}$$

where δ is the new bet amount and 5% is the maximum per-update shift.

8 Advanced Trading Features

8.1 Limit Orders

Users can place orders that execute only at target odds:

$$\text{LimitOrder} = \{m, s, \text{side}, \pi_{\text{target}}, t_{\text{expiry}}\} \tag{24}$$

A keeper monitors markets and calls `fill_limit_order()` when $\pi_{\text{current}} \leq \pi_{\text{target}}$.

8.2 Stop Loss

Automatic position exit when losing:

$$\text{Trigger Condition:} \quad \pi_{\text{your\_side}} < \theta_{\text{stop}} \tag{25}$$

where $\theta_{\text{stop}} \in [0.10, 0.90]$ is the user-specified threshold.

8.3 Partial Cashout

Exit early at current market odds:

$$\text{Payout} = (s_{\text{cashout}} - F) \times \pi_{\text{current}} \tag{26}$$

where $F = s_{\text{cashout}} \times \beta$ is the fee.

8.4 Conviction Betting

Lock bets for bonus payouts:

Table 6: Conviction Bonus Tiers

Lock Duration	Win Bonus
1 day	0.5%
7 days	3.5%
14 days	7.0%
30 days	15.0%

9 Social Trading Layer

9.1 Prediction Battles

Definition 9.1 (Battle)**.** A 1v1 battle is a pair of opposing bets:

$$\text{Battle} = \{m, s, \text{challenger}, \text{opponent}, \text{status}\} \tag{27}$$

where both parties stake equal amounts s on opposite sides.

Battle payout for winner:

$$\text{Payout} = 2s \times (1 - \beta_{\text{battle}}) \tag{28}$$

where $\beta_{\text{battle}} = 0.025$ (2.5%).

9.2 Guild System

Guilds enable pooled betting with profit sharing:

Definition 9.2 (Guild Treasury)**.** A guild maintains a pooled treasury:

$$T = \sum_{i=1}^{n} c_i \tag{29}$$

where c_i is member i's contribution.

Property 9.1 (Profit Distribution)**.** When a guild bet wins with profit Π:

$$\Pi_{\text{leader}} = 0.10 \times \Pi \tag{30}$$

$$\Pi_i = 0.90 \times \Pi \times \frac{c_i}{T} \quad \text{for member } i \tag{31}$$

9.3 Referral System

Referrers earn perpetual fee share:

$$R_{\text{referrer}} = 0.05 \times F_{\text{stakers}} \quad \text{for each referred user bet} \tag{32}$$

9.4 Leaderboards and Seasons

Predictor statistics are tracked on-chain:

$$\text{Accuracy} = \frac{n_{\text{correct}}}{n_{\text{total}}} \times 100\% \tag{33}$$

Seasons run for 30 days with prize pool distribution to top performers.

10 StableSwap AMM

10.1 Purpose

The StableSwap AMM provides near-zero slippage swaps between BAGS-IDL and PUMP-IDL tokens, which should trade at 1:1 parity.

10.2 Curve StableSwap Invariant

The AMM uses the Curve StableSwap invariant:

$$An^n \sum_i x_i + D = ADn^n + \frac{D^{n+1}}{n^n \prod_i x_i} \tag{34}$$

where:

- A = Amplification coefficient (1000)
- n = Number of tokens (2)
- x_i = Token balances [BAGS, PUMP]
- D = Invariant (total value proxy)

Theorem 10.1 (Slippage Reduction)**.** *For balanced pools, StableSwap provides $O(1/A)$ slippage compared to $O(1)$ for constant product AMMs.*

Table 7: Slippage Comparison: 1M Swap in 100M Pool

AMM Type	Output	Slippage
Constant Product	990,099	0.99%
StableSwap ($A = 100$)	999,800	0.02%
StableSwap ($A = 1000$)	999,960	0.004%

10.3 LP Token Economics

$$\text{LP APY} = \frac{V_{\text{daily}} \times 365 \times 0.001337 \times 0.50}{\text{TVL}} \times 100\% \tag{35}$$

where 0.001337 is the swap fee rate (13.37 bps) and 50% goes to LPs.

11 On-Chain Metrics Oracle

11.1 Design Constraints

The oracle operates under strict constraints, using only pure Solana RPC methods:

Property 11.1 (Pure RPC Oracle)**.** All metric calculations use only:

- `getAccountInfo(pubkey)`
- `getProgramAccounts(programId, filters)`
- `getMultipleAccounts(pubkeys[])`
- `getTokenAccountsByOwner(owner, filter)`
- `getSignaturesForAddress(address, options)`
- `getTransaction(signature)`

No third-party APIs (DeFiLlama, Pyth, etc.) are used.

This constraint ensures decentralization, censorship resistance, and verifiability.

11.2 TVL Calculation

Definition 11.1 (Total Value Locked)**.** For protocol P with vaults $\{V_1, \ldots, V_n\}$:

$$\mathrm{TVL}(P) = \sum_{i=1}^{n} \mathrm{balance}(V_i) \times \mathrm{price}(\mathrm{token}(V_i)) \tag{36}$$

Algorithm 1 TVL Computation via Pure RPC

Require: Protocol program ID P, vault PDAs $\{V_1, \ldots, V_n\}$
Ensure: TVL in USD

tvl $\leftarrow 0$
accounts $\leftarrow$ `getMultipleAccounts`($[V_1, \ldots, V_n]$)
for each (V_i, data) in accounts **do**
    balance $\leftarrow$ `parseTokenAccount`(data)
    mint $\leftarrow$ `extractMint`(data)
    price $\leftarrow$ `getPriceFromPool`(mint)
    tvl $\leftarrow$ tvl + balance $\times$ price
end for
return tvl

11.3 Volume Calculation: Stratified Sampling

Volume calculation faces a pagination limit of 1000 signatures per query. For high-volume protocols, we employ stratified sampling:

Definition 11.2 (Stratified Volume Estimator)**.** Partition the 24-hour window into k strata $\{S_1, \ldots, S_k\}$. For each stratum S_j:

$$\hat{V}_j = \frac{V_{\mathrm{sampled},j}}{r_j} \tag{37}$$

where $r_j = n_{\mathrm{sampled},j} / n_{\mathrm{total},j}$ is the sampling ratio.

Theorem 11.1 (Unbiased Estimation). *The stratified estimator $\hat{V} = \sum_j \hat{V}_j$ is unbiased:*

$$\mathbb{E}[\hat{V}] = V_{true} \tag{38}$$

Proof. For each stratum S_j, the sample mean is an unbiased estimator of the stratum mean. By linearity of expectation:

$$\mathbb{E}[\hat{V}] = \sum_j \mathbb{E}[\hat{V}_j] = \sum_j V_j = V_{\text{true}} \tag{39}$$

□

11.4 Complexity Analysis

Table 8: Algorithm Complexity

Operation	**Time**	**Space**	**RPC Calls**
TVL (n vaults)	$O(n)$	$O(n)$	$\lceil n/100 \rceil$
Volume (m txs, k strata)	$O(m)$	$O(k)$	$\lceil m/1000 \rceil$
Users (HyperLogLog)	$O(m)$	$O(1)$	$\lceil m/1000 \rceil$
Price (p pools)	$O(p)$	$O(p)$	p

For a typical protocol with 50 vaults and 100,000 daily transactions:

- TVL: 1 RPC call, <100ms latency
- Volume: 100 RPC calls, ∼5 minutes with rate limiting
- Users: 100 RPC calls, 16KB memory for HyperLogLog

The 95% confidence interval is:

$$\text{CI}_{95} = \hat{V} \pm 1.96\sqrt{\sum_j \frac{\sigma_j^2}{n_j}} \tag{40}$$

11.5 User Count: HyperLogLog

Counting unique users requires deduplication across millions of transactions. We use the HyperLogLog probabilistic data structure:

Definition 11.3 (HyperLogLog Estimator). The cardinality estimate is:

$$\hat{n} = \alpha_m \cdot m^2 \cdot \left(\sum_{j=1}^{m} 2^{-M_j}\right)^{-1} \tag{41}$$

where $m = 2^{14}$ is the register count, M_j is register j's value, and $\alpha_m \approx 0.7213$ is a bias correction constant.

Property 11.2 (HyperLogLog Error Bound). Standard error is $\sigma \approx 1.04/\sqrt{m} \approx 0.81\%$ for $m = 2^{14}$.

Memory usage is only 16 KB for arbitrarily large user sets.

11.6 Price Discovery

On-chain price discovery uses TWAP from liquidity pools:

Definition 11.4 (Time-Weighted Average Price)**.**

$$\text{TWAP}(t_0, t_1) = \frac{\sum_i p_i \times \Delta t_i}{\sum_i \Delta t_i} \tag{42}$$

where p_i is the price observation and Δt_i is the time weight.

Multi-pool aggregation for robustness:

$$p_{\text{final}} = \frac{\sum_{\text{pool}} p_{\text{pool}} \times L_{\text{pool}}}{\sum_{\text{pool}} L_{\text{pool}}} \tag{43}$$

where L_{pool} is liquidity depth (used as confidence weight).

11.7 Snapshot Consistency

Multi-slot consensus ensures data consistency:

Definition 11.5 (Multi-Slot Consensus)**.** Metrics are computed over a window $[s_{\text{start}}, s_{\text{end}}]$ where $s_{\text{end}} - s_{\text{start}} \leq 10$ slots. A result is valid if:

$$\frac{|\{s : \text{valid}(s)\}|}{s_{\text{end}} - s_{\text{start}}} \geq 0.8 \tag{44}$$

11.8 Confidence Scoring

Each resolution receives a confidence score:

$$C = w_1 C_{\text{data}} + w_2 C_{\text{price}} + w_3 C_{\text{freshness}} + w_4 C_{\text{coverage}} \tag{45}$$

where weights $(w_1, w_2, w_3, w_4) = (0.30, 0.30, 0.20, 0.20)$.

Table 9: Confidence-Based Actions

Confidence	**Action**
$C \geq 0.90$	Resolve immediately
$0.80 \leq C < 0.90$	Resolve with LOW_CONFIDENCE flag
$0.60 \leq C < 0.80$	Delay 1 hour, retry
$C < 0.60$	Cancel market, refund bets

12 Security Analysis

12.1 Smart Contract Security

The protocol implements multiple security measures:

- **Commit-Reveal:** Prevents front-running of bets and resolutions.
- **Oracle Bonding:** Economic accountability for resolution accuracy.

- **48-Hour Timelock:** Authority actions require waiting period.
- **Circuit Breaker:** Protocol can be paused in emergencies.
- **TVL Caps:** Gradual rollout limits exposure.
- **Insurance Fund:** Covers potential exploits.
- **Checked Arithmetic:** Prevents overflow/underflow.

12.2 Timing Constants

Table 10: Security Timing Parameters

Parameter	Value
MIN_RESOLUTION_DELAY	24 hours
BETTING_CLOSE_WINDOW	1 hour
BET_COMMIT_WINDOW	5 minutes
BET_REVEAL_WINDOW	1 hour
ORACLE_DISPUTE_WINDOW	1 hour
AUTHORITY_TIMELOCK	48 hours
MIN_STAKE_DURATION	24 hours

12.3 Attack Vector Analysis

Table 11: Attack Vectors and Mitigations

Attack	Severity	Mitigation
Oracle Manipulation	High	Bonding, slashing, dispute window
Front-Running	High	Commit-reveal scheme
Flash Loan	Medium	24h minimum stake duration
Contract Bugs	High	Audits, pausability, insurance
Governance Attacks	Medium	Timelock, veIDL distribution

12.4 Flash Loan Price Manipulation

Definition 12.1 (Flash Loan Defense)**.** Price readings are rejected if:

$$\frac{|p_{\text{spot}} - p_{\text{TWAP}}|}{p_{\text{TWAP}}} > 0.20 \tag{46}$$

where p_{TWAP} is computed over minimum 10 slots.

12.5 Sandwich Attack Prevention

Resolution uses VRF-delayed execution:

$$s_{\text{resolve}} = s_{\text{commit}} + \text{VRF}(100, 200) \tag{47}$$

where the resolution slot is unpredictable within the 100–200 slot range.

13 Governance

13.1 Vote-Escrow Mechanism

Voting power is determined by locked stake duration:

Definition 13.1 (veIDL Calculation)**.**

$$\text{veIDL} = s \times \frac{\Delta t}{4 \text{ years}} \tag{48}$$

where s is staked IDL and Δt is lock duration.

Property 13.1 (Linear Decay)**.** veIDL decreases linearly as lock expires:

$$\text{veIDL}(t) = \text{veIDL}_0 \times \frac{t_{\text{end}} - t}{t_{\text{end}} - t_{\text{start}}} \tag{49}$$

13.2 Proposal Lifecycle

1. **Discussion:** 3 days for community input.
2. **Voting:** 5 days for veIDL holders to vote.
3. **Timelock:** 2 days before execution.
4. **Execution:** Proposal enacted on-chain.

Requirements:

- **Quorum:** 20% of veIDL supply must participate.
- **Majority:** 50%+1 of votes cast.

13.3 Governable Parameters

Table 12: Governance-Controlled Parameters

Parameter	**Current**	**Range**	**Description**
BET_FEE_BPS	300	100–500	Fee on winning bets
STAKER_FEE_SHARE	50%	30–70%	Staker portion
BURN_FEE_SHARE	10%	5–20%	Burn portion
MIN_BET_AMOUNT	0.001	0.001–1	Minimum bet

14 Limitations and Future Work

14.1 Current Limitations

We acknowledge several limitations of the current design:

1. **RPC Reliability:** The pure-RPC oracle depends on RPC node availability. While multi-source validation mitigates this, a coordinated RPC outage during resolution could delay markets.

2. **Closed-Source Protocols:** Protocols without published IDLs require heuristic balance estimation, reducing confidence scores. Approximately 20% of top-100 Solana protocols lack public IDLs.
3. **Historical Data:** Solana RPC provides no historical state queries. Trend-based predictions (e.g., "TVL increases 10%") require off-chain snapshot databases, introducing trust assumptions.
4. **Volume Estimation Variance:** For extremely high-volume protocols (>1M daily transactions), stratified sampling introduces 2–5% estimation error due to the 1000-signature pagination limit.
5. **Cross-Chain Limitations:** The oracle cannot verify assets bridged from other chains. Bridge exploits may not be detected until significant price deviation occurs.
6. **Governance Centralization:** Initial protocol deployment requires trusted setup. Full decentralization depends on sufficient veIDL distribution, which takes time to develop.

14.2 Future Work

Several directions for future research and development:

- **Zero-Knowledge Proofs:** ZK-SNARKs could enable verifiable off-chain computation of complex metrics, reducing on-chain costs while maintaining trustlessness.
- **Cross-Chain Expansion:** Extend the IDL registry and prediction markets to EVM chains, enabling cross-ecosystem metric comparisons.
- **Machine Learning Integration:** AI-powered market makers could provide better initial odds estimation and liquidity bootstrapping.
- **Continuous Prediction Markets:** Move beyond binary outcomes to scalar predictions with continuous payoff functions.
- **Formal Verification:** Apply formal methods to verify smart contract correctness, particularly for the commit-reveal and payout logic.

15 Conclusion

IDL Protocol presents a comprehensive solution for prediction markets on Solana, uniquely combining:

1. A free, open IDL registry serving as infrastructure for the Solana ecosystem.
2. A prediction market system with provably fair parimutuel payouts and front-running resistance.
3. An on-chain metrics oracle that operates without third-party dependencies, using novel algorithms for TVL, volume, and user count computation.
4. Social trading features that enhance engagement and create network effects.
5. A deflationary token model with real yield from protocol revenue.

The protocol's reliance on pure RPC data sources, while technically challenging, provides strong decentralization and censorship resistance guarantees that are essential for a trustworthy prediction market infrastructure.

Acknowledgments

We thank the Solana and Anchor development communities for their foundational work, and the early IDLHub users who contributed to the registry.

A Contract Addresses

Table 13: Deployed Contract Addresses

Contract	Address
Devnet	
IDL Protocol	`BSn7neicVV2kEzgaZmd6tZEBm4tdgzBRyELov65Lq7dt`
IDL StableSwap	`EFsgmpbKifyA75ZY5NPHQxrtuAHHB6sYnoGkLi6xoTte`
Token Mints	
PUMP-IDL	`4GihJrYJGQ9pjqDySTjd57y1h3nNkEZNbzJxCbispump`
BAGS-IDL	`8zdhHxthCFoigAGw4QRxWfXUWLY1KkMZ1r7CTcmiBAGS`

B Glossary

Term	Definition
IDL	Interface Definition Language—JSON schema describing Solana program interfaces
veIDL	Vote-escrowed IDL—locked staking tokens with governance power
MCP	Model Context Protocol—AI agent API standard
Parimutuel	Betting system where all bets pooled, winners split loser pool
TVL	Total Value Locked—assets deposited in a protocol
Commit-Reveal	Two-phase scheme preventing front-running
StableSwap	AMM optimized for pegged assets (Curve-style)
RPC	Remote Procedure Call—API for querying Solana blockchain state
TWAP	Time-Weighted Average Price—price averaged over time
HyperLogLog	Probabilistic algorithm for counting unique elements
PDA	Program Derived Address—deterministic account addresses
CLMM	Concentrated Liquidity Market Maker

References

[1] Yakovenko, A. (2018). Solana: A new architecture for a high performance blockchain. *Solana Whitepaper.*

[2] Armani, A. (2021). Anchor: A framework for Solana programs. https://anchor-lang.com

[3] Egorov, M. (2019). StableSwap—efficient mechanism for Stablecoin liquidity. *Curve Finance Whitepaper.*

[4] Flajolet, P., Fusy, É., Gandouet, O., & Meunier, F. (2007). HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete Mathematics and Theoretical Computer Science*.

[5] Cronje, A. (2022). ve(3,3): Tokenomics with vote-escrowed tokens. *Solidly Protocol Whitepaper*.

[6] Eisenberg, E. (1959). Aggregation of utility functions. *Management Science*, 7(4), 337–350.

[7] Forsythe, R., Nelson, F., Neumann, G. R., & Wright, J. (1992). Anatomy of an experimental political stock market. *The American Economic Review*, 82(5), 1142–1161.

[8] Peterson, J., Krug, J., Zoltu, M., Williams, A. K., & Alexander, S. (2015). Augur: a decentralized oracle and prediction market platform. *arXiv preprint arXiv:1501.01042*.

[9] Hanson, R. (2003). Combinatorial information market design. *Information Systems Frontiers*, 5(1), 107–119.

[10] Arrow, K. J., Forsythe, R., Gorham, M., Hahn, R., Hanson, R., Ledyard, J. O., ... & Zitzewitz, E. (2008). The promise of prediction markets. *Science*, 320(5878), 877–878.

[11] Ellis, S., Juels, A., & Nazarov, S. (2017). ChainLink: A decentralized oracle network. *Chainlink Whitepaper*.

[12] Adams, H., Zinsmeister, N., & Robinson, D. (2020). Uniswap v2 core. *Uniswap Whitepaper*.

[13] Blum, M. (1983). Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1), 23–27.