

The Truth-It Network: A Decentralized Binary Oracle with Reputation-Weighted Commit-Reveal Voting

Lukas Saul*

Robert Gludo[†]

Jonathan Gludo[‡]

September 28, 2026

Abstract

The Truth-It Network is a Solana-native binary oracle protocol. It allows a permissioned network of truth providers to answer yes/no questions through a commit-reveal voting scheme, with vote weight determined by an on-chain reputation score. Rewards are held in per-question vaults and distributed proportionally to winning vote weight. This whitepaper describes the protocol’s membership model, question lifecycle, reward distribution, rent reclamation, and reputation calculation.

1 Introduction: From *EarthWeb* to On-Chain Oracles

In Marc Stiegler’s 1999 novel *EarthWeb*, humanity defends itself against periodic alien assaults by consulting a global prediction market called “castpoints.” Trained operatives aboard alien vessels receive guidance derived from the aggregated forecasts of millions of participants, each wagering on the internal layout of the ship they are about to infiltrate. The novel’s central insight—that a decentralized network of self-interested forecasters can produce more reliable truth than any centralized authority—anticipated by two decades the emergence of blockchain-based oracle systems.

This literary lineage extends further back. Jules Verne’s *Around the World in Eighty Days* (1872) centers on a wager that functions as a proto-prediction market: Phileas Fogg’s twenty-thousand-pound bet on his own circumnavigation creates a financial instrument whose value fluctuates with news and public sentiment, traded on the London Exchange as “Fogg bonds”. In Chinese science fiction, Xiao Xinghan’s *Alea Aacta Est (The Die Has Been Cast)* (骰子已掷出, 2021) depicts a world governed by the “Sacred Order,” a centralized algorithmic system that predicts and controls all human behavior, while a casino’s “Spider” system reads micro-expressions to forecast gamblers’ actions—a dark mirror of the prediction market’s goal of aggregating information to forecast outcomes.

The technical history of blockchain oracles is itself now a subject of study. Caldarelli traces the lineage from early Bitcoin “reality keys” through Oraclize, documenting how the oracle problem evolved from a practical data-feed concern into a foundational design challenge for smart contracts [1]. In a companion survey, Caldarelli extends this analysis to the pre-Ethereum era, showing that the architectural patterns of modern oracles—trusted reporters, consensus aggregation, and

*Hefei University of Technology 合肥工业大学, Vermont Secure Computing Consultancy

[†]Vermont Secure Computing Consultancy

[‡]Vermont Secure Computing Consultancy

cryptographic attestation—were present well before the term “oracle” entered common use [4]. The Tokenomics 2019 proceedings frame these developments within the broader economic and security questions raised by blockchain protocols, including the incentive design of decentralized agreement mechanisms [2]. Park’s master’s thesis proposes a data-retailer model for decentralized oracle design, emphasizing the importance of verifiable data provenance [3]. More recently, Yao et al. introduce PSCBO, a provably secure consensus-based oracle, advancing formal guarantees for oracle correctness [5]. And Augur’s original whitepaper remains the canonical reference for decentralized prediction markets, describing a token-based mechanism for resolving real-world events [6].

The Truth-It Network is a Solana-native implementation of the decentralized insight. It is a binary oracle: a protocol for posing yes/no questions to a decentralized network of truth providers, who commit and reveal votes under a commit-reveal scheme, weighted by a reputation score earned through prior correct participation. The protocol is designed around three principles: Sybil resistance through invitation, truth discovery through skin-in-the-game incentives, and deterministic settlement through on-chain finalization.

2 Protocol Overview

The Truth-It Network allows any participant to:

1. **Join the network**—either through one of 33 bootstrap slots or via invitation from an existing member.
2. **Create a binary question**—with a reward pool of at least 0.05 SOL, a commit deadline, and a reveal deadline.
3. **Commit a vote**—by submitting a cryptographic hash of the chosen option and a secret password.
4. **Reveal the vote**—after the commit phase closes, by submitting the password that reproduces the commitment.
5. **Claim rewards**—if the voter chose the winning option and the question meets minimum participation thresholds.
6. **Earn reputation**—which increases vote weight in future questions and unlocks invite tokens.
7. **Reclaim rent**—for losing or non-revealing voters after the reveal phase ends.
8. **Drain unclaimed rewards**—to the protocol fee receiver after a 30-day claim window expires.

The protocol is implemented as a single Anchor program on Solana. All state is stored in program-derived accounts (PDAs), and all reward custody is handled by per-question vault accounts.

3 Membership and Sybil Resistance

The Truth-It Network does not permit open, permissionless joining. Instead, membership expands through a two-tier mechanism:

- **Bootstrap phase.** The first 33 participants may join without invitation. This cap is enforced by a `GlobalState` account tracking `truth_provider_count`. The number 33 is arbitrary but intentionally small, ensuring that the initial truth provider set is known and accountable.
- **Invitation phase.** After the bootstrap slots are filled, new members must present a valid

Invite PDA. Invitations are created by existing members who have earned `invite_tokens` through correct voting. An invite is bound to a specific nominee (`invite.invitee`), and the nominee’s `UserRecord` PDA is derived from their public key. This prevents front-running or invite theft.

A `MembershipRecord` account tracks whether a user has ever joined, allowing returning members to rejoin without consuming a new bootstrap slot or requiring a fresh invitation. This design preserves the scarcity of the initial truth provider set while allowing voluntary exits and returns.

4 Question Lifecycle

4.1 Creation

Any network member may create a question by calling `create_question`. The question must satisfy:

- **Text length:** 10–150 characters.
- **Reward:** at least 0.05 SOL (50,000,000 lamports), transferred from the asker to a per-question vault PDA.
- **Timing:** `now < commit_end_time < reveal_end_time`.

The question is assigned a monotonically increasing `id` from the asker’s `QuestionCounter` PDA. The question PDA is derived from `[b"question", asker, id]`, and the vault PDA from `[b"vault", question]`. Both are initialized in the same transaction. Questions are binary: `option_1` = “True” and `option_2` = “False”.

4.2 Commit Phase

During the commit phase (`now < commit_end_time`), members may submit a 32-byte commitment:

```
commitment = keccak256("truth-vote-v1" || question_key || voter_key || vote_byte || password)
```

where `vote_byte` $\in \{1, 2\}$. The commitment is stored in a `VoterRecord` PDA derived from `[b"vote", voter, question]`. The voter’s `UserRecord` must exist, and the voter must not have already committed to this question. Committing does not reveal the vote; the password is known only to the voter.

4.3 Reveal Phase

After `commit_end_time` and before `reveal_end_time`, voters may reveal by submitting the password. The program iterates over both possible votes, recomputes the hash, and checks it against the stored commitment. If a match is found, the vote is recorded and weighted:

$$\text{vote_weight} = \max(1, \text{user_record.reputation})$$

Reputation is a `u8` from 1 to 19, derived from the sum of `total_revealed_votes` and `total_correct_votes` via the `calculate_reputation` function (Section 6). A voter with reputation 0 (a new member who has never voted) has weight 1. A voter with reputation 19 has weight 19. The question’s `votes_option_1` or `votes_option_2` is incremented by the vote weight, and `revealed_voters_count` is incremented.

4.4 Finalization

Finalization may be triggered by any of several instructions: `finalize_voting`, `claim_reward`, `drain_unclaimed_reward`, or `reclaim_commit_or_loser_rent`. The `ensure_question_finalized` helper computes:

$$\begin{aligned} \text{total_votes} &= \text{votes_option_1} + \text{votes_option_2} \\ \text{winning_option} &= \begin{cases} 0 & \text{if } \text{total_votes} = 0, \\ 0 & \text{if } \text{votes_option_1} = \text{votes_option_2}, \\ 1 & \text{if } \text{votes_option_1} > \text{votes_option_2}, \\ 2 & \text{otherwise.} \end{cases} \end{aligned}$$

`eligible_voters` is set to the total weight of the winning option (or total votes in a tie). The question is marked `finalized = true`. Finalization is idempotent and permissionless: anyone may call it, and calling it multiple times has no effect beyond the first.

5 Reward Distribution

5.1 Fee and Snapshot

The first claim on a finalized question triggers a one-time snapshot:

- The vault balance is read, and the minimum rent-exempt balance for the vault account is subtracted.
- A **2% fee** is taken from the available reward and transferred to `FEE_RECEIVER_PUBKEY`.
- The remaining `snapshot_reward` is recorded, along with `snapshot_total_weight` (the total weight of the winning option, or both options in a tie).

This snapshot ensures that the reward per unit weight is fixed at the moment of first claim, preventing later claimers from diluting earlier ones.

5.2 Claim Calculation

For each eligible voter (a voter who chose the winning option, or any revealed voter in a tie), the base share is:

$$\text{base_share} = \frac{\text{snapshot_reward} \times \text{vote_weight}}{\text{snapshot_total_weight}}$$

The **last claimer** (the one whose `claimed_weight` reaches `snapshot_total_weight`) receives the remaining undistributed reward, correcting for integer rounding. This guarantees that the full snapshot reward is distributed. If the vault balance is insufficient to cover a share, the share is capped at the available balance. If the vault is drained before all eligible voters claim, later claimers receive nothing.

6 Reputation Calculation

Reputation is the protocol's mechanism for weighting votes and gating invitation privileges. It is a `u8` value in the range `[1, 19]`, computed deterministically from two counters stored in each user's `UserRecord`:

- `total_revealed_votes`—the number of qualifying votes the user has revealed, regardless of outcome.
- `total_correct_votes`—the number of qualifying votes the user has revealed that matched the winning option.

6.1 Qualifying Votes

A revealed vote qualifies for reputation only if the question satisfies two conditions at the time of claim or reclamation:

1. `revealed_voters_count` ≥ 3 , and
2. `reveal_end_time` $-$ `created_at` $\geq 86,400$ seconds (one day).

These thresholds prevent reputation farming through trivial or low-participation questions. A voter who commits but does not reveal receives no reputation change. A voter who reveals a losing option increments `total_revealed_votes` but not `total_correct_votes`.

6.2 The Reputation Function

Let

$$s = \text{total_revealed_votes} + \text{total_correct_votes}.$$

The reputation function $R(s)$ is a step function mapping s to a `u8` in $[1, 19]$. Its progression resembles the leveling curve of a role-playing game: early participation yields rapid gains, but each subsequent level requires progressively more experience, with the step widths growing roughly geometrically. A user who has revealed one vote reaches level 1; ten qualifying votes correspond to level 8; and the maximum level of 19 is not attainable until $s = 309$, a threshold that demands sustained, long-term accuracy. Table 1 enumerates the mapping.

The maximum reputation is 19. Vote weight is then $\max(1, R(s))$, so a user with $s = 0$ or 1 has weight 1, and a user at the cap has weight 19.

6.3 Invite Token Accrual

Reputation also governs invitation power. When a user casts a qualifying correct vote and currently holds no unused invite token (`invite_tokens` = 0), the protocol increments `invite_correct_votes`. When that counter reaches 3, the user earns one `invite_token` and the counter resets to 0. A user may hold at most one unused invite token at a time; further correct votes do not accrue progress until the token is spent via `nominate_invitee`. This design throttles network growth to the rate at which existing members demonstrate sustained accuracy.

7 Avoiding the Pitfalls of Token-Weighted Oracles

A significant class of existing decentralized oracles, including UMA’s Optimistic Oracle as used by Polymarket, resolves disputes through token-weighted voting. In these systems, the weight of a vote is proportional to the voter’s stake in a governance token (e.g., UMA’s REP or the UMA token itself), irrespective of the voter’s track record or expertise. This design introduces a structural vulnerability: large token holders can influence outcomes to protect their own financial positions in the markets being resolved. A widely reported dispute in 2026 over an \$80 million Polymarket market on MicroStrategy’s Bitcoin sales illustrated this concern, with critics arguing

s	$R(s)$
0 or 1	1
2	2
3	3
4	4
5	5
6	6
7–9	7
10–12	8
13–17	9
18–25	10
26–34	11
35–46	12
47–62	13
63–87	14
88–118	15
119–163	16
164–224	17
225–308	18
309 and above	19

Table 1: Reputation as a function of $s = \text{total_revealed_votes} + \text{total_correct_votes}$. The step widths approximate an RPG-style leveling curve.

that concentrated voting power skewed the resolution away from the market’s stated rules and toward the interests of major stakeholders. Galaxy Research, a participant in that market, explicitly criticized the model, stating that “prediction markets should price what happens, not how the oracle will reinterpret rules after the fact.”

The Truth-It Network avoids this class of problem through a fundamentally different weighting mechanism. Vote weight is not proportional to capital stake, but to an earned reputation that can only increase through demonstrated accuracy over time. A user cannot acquire reputation by purchasing tokens; they must participate, reveal, and be correct, under conditions that require minimum participation and duration. This design shifts the incentive from “what do the largest token holders want?” to “what does the most accurate track record say?” The commit-reveal scheme further protects against manipulation by ensuring that votes are sealed until after the commit deadline, preventing last-minute herding or retaliation. The result is a system whose settlement mechanism is deterministic, rule-bound, and resistant to the whale-dominated governance critiques leveled at token-weighted alternatives. This approach aligns with the economic and security concerns raised in the Tokenomics 2019 proceedings regarding incentive-compatible protocol design [2], and with the formal correctness goals of consensus-based oracle constructions such as PSCBO [5]. Unlike the token-curated registry model implicit in Augur’s design [6], the Truth-It Network does not require a speculative asset to mediate truth; reputation is earned, not bought.

8 Rent Reclamation and Cleanup

Three mechanisms return rent to participants and close accounts:

8.1 `reclaim_commit_or_loser_rent`

After the reveal phase ends and the question is finalized, a voter who either did not reveal or revealed a losing option may reclaim their `VoterRecord` rent. The account is closed, and the voter receives the rent lamports. If the voter revealed, their reputation may be updated as described in Section 6.

8.2 `cleanup_expired_voter_record`

Anyone may clean up a `VoterRecord` that belongs to a non-revealing voter, after the reveal phase ends. The rent is returned to the voter. This allows the network to reclaim storage on behalf of inactive participants.

8.3 `delete_expired_question`

Thirty days after `reveal_end_time`, the asker may close the `Question` and `Vault` accounts, recovering their rent. This is permitted only if no one committed, or no votes were revealed, or rewards are fully distributed and all voter records are closed. The 30-day window is enforced by `CLAIM_EXPIRY_SECS`.

9 Draining Unclaimed Rewards

If a question is finalized but some eligible voters never claim, the remaining vault balance may be drained to the fee receiver after the 30-day claim window expires. The `drain_unclaimed_reward` instruction:

- Requires `now ≥ reveal_end_time + CLAIM_EXPIRY_SECS`.
- Requires the question to be finalized (or finalizes it if revealed votes exist).
- Transfers the vault balance minus rent to `FEE_RECEIVER_PUBKEY`.
- Sets `reward_drained = true`.

This prevents indefinite lockup of unclaimed rewards while preserving the claim window for diligent participants.

10 Security Properties

The protocol’s security guarantees arise from the interaction of its membership model, commit-reveal scheme, reputation function, and reward custody rules. Table 2 summarizes the principal properties and the mechanism that enforces each.

11 Parameters

The protocol’s behavior is governed by a small set of constants, listed in Table 3. These values are fixed at deployment and are not adjustable by any on-chain authority.

12 Conclusion

The Truth-It Network is a minimal, self-contained binary oracle. It does not attempt to resolve arbitrary questions or integrate with external data sources. Instead, it provides a cryptoeconomic

Property	Mechanism
Sybil resistance	Invitation-only membership after 33 bootstrap slots; invite bound to nominee.
Vote secrecy	Commit-reveal with keccak256 hash; password never stored on-chain.
Vote integrity	Commitment includes question key and voter key, preventing replay across questions or voters.
Reputation integrity	Reputation only increases through qualifying revealed votes; minimum 3 voters and 1-day duration.
Reward solvency	Snapshot at first claim; last-claimer remainder; vault balance checked before each transfer.
Rent safety	Rent-exempt minimum preserved in vault; reclaim and cleanup paths return rent to participants.
Idempotent finalization	<code>ensure_question_finalized</code> is safe to call multiple times.
Expiry safety	30-day claim window; drain only after expiry; delete only after settlement.

Table 2: Security properties and their enforcing mechanisms.

Parameter	Value	Notes
Bootstrap slots	33	Initial truth providers.
Minimum reward	0.05 SOL	50,000,000 lamports.
Protocol fee	2%	Taken from available reward at first claim.
Claim expiry	30 days	After <code>reveal_end_time</code> .
Minimum voters for reputation	3	Revealed voters.
Minimum duration for reputation	86,400 s	One day.
Reputation range	1–19	u8, derived from revealed + correct votes.
Invite threshold	3 correct votes	Earns one invite token.
Vote weight	$\max(1, \text{reputation})$	Reputation 0 $\rightarrow$ weight 1.

Table 3: Protocol parameters.

primitive: a commit-reveal vote with reputation-weighted outcomes and invitation-gated membership. Its design is inspired by the prediction markets of science fiction, but its implementation is grounded in the constraints and affordances of Solana: cheap transactions, PDA-derived state, and Anchor’s account validation.

The protocol is intentionally conservative. It caps bootstrap membership, requires minimum participation for reputation, and enforces a 30-day claim window. These choices prioritize Sybil resistance and deterministic settlement over open accessibility. For applications that require a trusted binary signal—governance votes, dispute resolution, conditional escrow—the Truth-It Network offers a verifiable, on-chain mechanism for aggregating decentralized judgment.

References

- [1] Giulio Caldarelli. From Reality Keys to Oraclize: A Deep Dive into the History of Bitcoin Oracles. *arXiv preprint arXiv:2302.07911*, Feb. 2023. <https://arxiv.org/abs/2302.07911>
- [2] Vincent Danos, Maurice Herlihy, Maria Potop-Butucaru, Julien Prat, and Sara Tucci-Piergiovanni (eds.). *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*. OASICs, Vol. 71, Schloss Dagstuhl, Paris, France, May 2019. <https://doi.org/10.4230/OASICs.Tokenomics.2019.0>
- [3] Soobin Park. *A Data Retailer Based Decentralized Oracle Design for Smart Contracts in Blockchain*. Master’s thesis, Seoul National University, Feb. 2019. <https://s-space.snu.ac.kr/handle/10371/150787>
- [4] Giulio Caldarelli. Before Ethereum: The Origin and Evolution of Blockchain Oracles. *IEEE Access*, 11:50899–50917, May 2023. <https://doi.org/10.1109/ACCESS.2023.3276903>
- [5] Yusong Yao, Yiwen Shi, Guohua Tian, Meixia Miao, and Willy Susilo. PSCBO: A Provably Secure Consensus-Based Blockchain Oracle. *Computer Standards & Interfaces*, 91, 2025. <https://doi.org/10.1016/j.csi.2024.103892>
- [6] Jack Peterson and Joseph Krug. Augur: A Decentralized, Open-Source Platform for Prediction Markets. *arXiv preprint arXiv:1501.01042*, 2015. <https://arxiv.org/abs/1501.01042>

This whitepaper describes the Truth-It Network program as deployed at the program ID declared in the source. It is not financial advice. Participants should review the source code and security policy before interacting with the protocol.