# Notch Protocol

A Cryptographic Framework for Verifiable Prediction Scoring and Alpha Commoditization

**Qais Alassa**
Bethlehem, Palestine

**Osama Alashqar**
Oslo, Norway

---

Technical Specification v0.1
March 2026

notch.finance

## Abstract

We present NOTCH, a protocol for the cryptographic commitment, verification, and scoring of price predictions on Ethereum Layer 2. The protocol implements a commit-reveal scheme in which a predictor submits a hashed prediction to a smart contract, reveals the prediction parameters after a mandatory delay, and receives an automated accuracy score upon resolution by an oracle price feed. Scoring uses the Brier Score—a strictly proper scoring rule with the property that expected score is uniquely optimized when stated confidence equals true belief, making systematic misreporting irrational under all strategies. Over a sequence of resolved predictions, each participant accumulates a composite reputation metric (the *Notch Score*) that is permanent, public, and verifiable on-chain.

The protocol introduces three financial instruments constructed on this scoring primitive: *Alpha Passes*, which are tokenized access rights to a predictor's future output; *Alpha Futures*, which are binary options on the trajectory of a predictor's score; and *Alpha Indices*, which bundle the output of top-ranked predictors into a single exposure. Together, these instruments create a market in which verified prediction skill becomes a tradeable commodity with continuous price discovery.

This document specifies the commitment scheme, the scoring mechanism and its game-theoretic properties, the instrument layer, the oracle and security architecture, and the token economics of the $NOTCH token. The protocol is designed for deployment on Arbitrum One, with a complete prediction cycle costing approximately $0.01–$0.03 in gas.



**Keywords:** prediction markets, proper scoring rules, commit-reveal, on-chain reputation, DeFi primitives, alpha commoditization

# 1 Introduction

The ability to predict price movements is the most valued and least verifiable skill in financial markets. A substantial industry exists around the distribution of trading intelligence—signal groups, copy-trading platforms, subscription newsletters, social media commentary—yet no widely adopted mechanism exists for verifying the accuracy of these claims against a cryptographic standard. The core problem is structural: in the absence of commitment before the fact, any post-hoc record of predictions is susceptible to selective deletion, retroactive fabrication, and survivorship curation.

Existing approaches address fragments of this problem without solving it. Prediction markets such as Polymarket [5] and Kalshi [6] price discrete event outcomes through open order books, achieving significant scale ($63.5B combined volume in 2025), but they evaluate *events*, not the skill of individual predictors—a trader who correctly prices one event carries no verifiable credential into the next. Copy-trading platforms such as Copin.io [7] observe and rank on-chain trading activity, but they operate retrospectively on executed trades rather than on committed predictions, and they capture neither stated confidence nor calibration quality. Crowdsourced intelligence systems such as Numerai [8] incentivize prediction accuracy through staking and tournament mechanics, but lock all output inside a single managed fund, preventing the formation of open markets for individual predictor performance.

NOTCH addresses this gap with a protocol that satisfies four properties simultaneously:

1. **Cryptographic commitment.** Predictions are hashed and recorded on-chain before the predicted event occurs. The commitment is binding (it cannot be altered after submission) and hiding (the prediction content is invisible until revealed).
2. **Calibration-aware scoring.** Accuracy is evaluated using a strictly proper scoring rule that rewards not only correctness but also the quality of stated confidence, penalizing both overconfidence and underconfidence.
3. **Persistent on-chain reputation.** Scores accumulate over time into a composite metric that is public, queryable by any smart contract, and resistant to manipulation through Sybil strategies.
4. **Financial instruments on verified skill.** The score serves as the underlying asset for tokenized access rights, derivative contracts, and index products, enabling price discovery for prediction quality.

No existing protocol combines all four properties. The remainder of this paper specifies each component in detail: the commitment scheme (§2), the scoring mechanism (§3), the instrument layer (§4), the oracle and security architecture (§5), the token economics (§6), and the implementation plan (§7).

# 2 Prediction Commitment Scheme

The commitment scheme is the cryptographic foundation of the protocol. It ensures that a prediction exists on-chain, in unalterable form, before the outcome it predicts is observable.

## 2.1 Commit-Reveal-Resolve Lifecycle

A prediction in NOTCH passes through four discrete states. We define the lifecycle formally.

**Definition 2.1** (Prediction). *A prediction $P$ is a tuple $(a, d, \tau, c, t_{\text{exp}})$ where $a \in \mathcal{A}$ is an asset identifier, $d \in \{\text{ABOVE}, \text{BELOW}\}$ is a directional call, $\tau \in \mathbb{R}^+$ is a target price, $c \in (0, 1)$ is the predictor's stated confidence, and $t_{\text{exp}} \in \mathbb{N}$ is the expiry timestamp.*

The lifecycle proceeds as follows:

**Phase 1: Commit.** The predictor constructs the prediction tuple $P$ and a random salt $s$, then computes:

$$h = \texttt{keccak256}\big(\texttt{abi.encodePacked}(a, d, \tau, c, t_{\text{exp}}, s)\big) \tag{1}$$

The hash $h$ is submitted to the `PredictionEngine` contract, which records $h$, the sender address, the current block timestamp $t_{\text{commit}}$, and the claimed expiry $t_{\text{exp}}$. The prediction transitions to the COMMITTED state.

The contract enforces:

- $t_{\text{exp}} - t_{\text{commit}} \geq \Delta_{\min}$, where $\Delta_{\min}$ is the minimum prediction horizon (a governance parameter; initially 15 minutes).
- $t_{\text{exp}} - t_{\text{commit}} \leq \Delta_{\max}$, where $\Delta_{\max}$ is the maximum prediction horizon (initially 30 days).
- The predictor has staked a minimum amount of \$NOTCH tokens.

**Phase 2: Reveal.** After the commit and before expiry, the predictor submits the full prediction tuple $(a, d, \tau, c, t_{\text{exp}}, s)$ in plaintext. The contract verifies:

$$\texttt{keccak256}\big(\texttt{abi.encodePacked}(a, d, \tau, c, t_{\text{exp}}, s)\big) \stackrel{?}{=} h \tag{2}$$

If the hash matches, the prediction transitions to REVEALED and the plaintext parameters are stored on-chain. If the predictor fails to reveal before $t_{\text{exp}}$, the prediction is marked EXPIRED and treated as a maximally incorrect prediction in the scoring function (see §3).

A mandatory delay $\delta_{\text{reveal}}$ is enforced between commit and reveal: the reveal transaction is rejected if $t_{\text{reveal}} - t_{\text{commit}} < \delta_{\text{reveal}}$. This prevents a strategy in which a predictor commits and immediately reveals to front-run market reaction.

**Phase 3: Resolve.** At or after $t_{\text{exp}}$, any address may call the `resolve()` function. The contract queries the oracle price feed for asset $a$ at timestamp $t_{\text{exp}}$ and obtains the settlement price $p^*$. The binary outcome is computed:

$$o = \begin{cases} 1 & \text{if } d = \text{ABOVE and } p^* > \tau, \\ 1 & \text{if } d = \text{BELOW and } p^* < \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The prediction transitions to RESOLVED and the outcome $o$ is recorded permanently.

**Phase 4: Score.** Upon resolution, the predictor's score is updated according to the scoring function defined in §3. The updated score is written to the `ScoreRegistry` contract. The prediction lifecycle is complete.

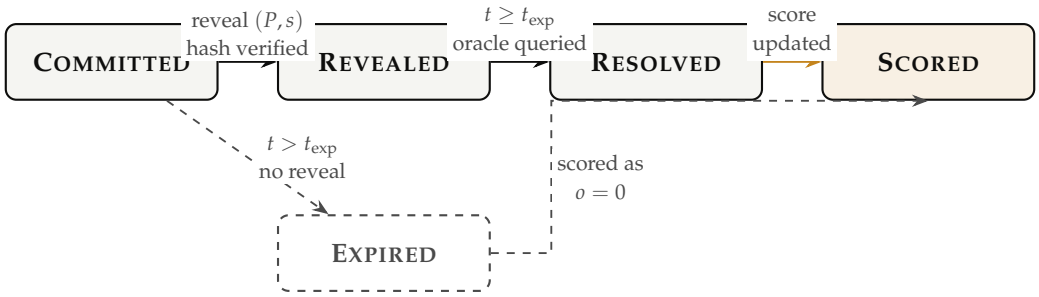


Figure 1: Prediction state machine. Solid transitions represent the normal lifecycle. Dashed transitions handle non-revelation, which is penalized as a maximally incorrect prediction.

### 2.2 State Machine

Figure 1 illustrates the four-state lifecycle with transition conditions.

### 2.3 Security Properties

The commitment scheme provides three guarantees:

**Property 2.1** (Hiding). *The commitment hash $h$ reveals no information about the prediction $P$. Since `keccak256` is a cryptographic hash function with pre-image resistance, an observer who sees $h$ in the mempool or on-chain cannot determine $a$, $d$, $\tau$, $c$, or $t_{\text{exp}}$. The salt $s$ prevents dictionary attacks against common prediction values.*

**Property 2.2** (Binding). *A predictor who has committed $h$ cannot produce a different prediction $P' \neq P$ that hashes to the same value. This follows from the collision resistance of `keccak256`: finding two distinct inputs with the same hash is computationally infeasible.*

**Property 2.3** (Non-repudiation). *A predictor cannot escape accountability for a committed prediction. If the prediction is revealed, it resolves and scores. If it is not revealed, it is scored as a failure. There is no third option. The "delete message" strategy that plagues unverified signal channels is structurally impossible.*

## 3 Scoring Mechanism

The scoring mechanism must satisfy a demanding set of requirements: it must reward accuracy, penalize miscalibration, resist gaming through confidence manipulation, and remain computationally feasible on-chain. We achieve this through a composite metric built on the Brier Score, a strictly proper scoring rule introduced by Brier [1] and extensively studied in the forecasting literature [2, 3].

### 3.1 The Brier Score and Strict Propriety

**Definition 3.1** (Brier Score). *For a sequence of $N$ predictions with stated confidences $f_1, \ldots, f_N \in [0, 1]$ and binary outcomes $o_1, \ldots, o_N \in \{0, 1\}$, the Brier Score is:*

$$\text{BS} = \frac{1}{N} \sum_{i=1}^{N} (f_i - o_i)^2 \tag{4}$$

*The Brier Score ranges from* 0 *(perfect calibration and accuracy) to* 1 *(maximal inaccuracy). Lower is better.*

The Brier Score belongs to the family of *strictly proper scoring rules*, which possess the following property:

**Proposition 3.1** (Strict Propriety)**.** *Let $p$ be the predictor's true belief about the probability of the outcome $o = 1$. The expected Brier Score $\mathbb{E}[(f - o)^2]$ is uniquely minimized when $f = p$. That is, for any $f \neq p$:*

$$\mathbb{E}_{o\sim\text{Bernoulli}(p)}[(f - o)^2] > \mathbb{E}_{o\sim\text{Bernoulli}(p)}[(p - o)^2] \tag{5}$$

*Sketch.* Expanding the expectation:

$$\begin{aligned}\mathbb{E}[(f - o)^2] &= p(f - 1)^2 + (1 - p)f^2 \\ &= pf^2 - 2pf + p + f^2 - pf^2 \\ &= f^2 - 2pf + p\end{aligned} \tag{6}$$

Differentiating with respect to $f$ and setting to zero: $2f - 2p = 0$, yielding $f^* = p$. The second derivative is $2 > 0$, confirming a unique minimum. Thus, the expected score is minimized if and only if the predictor reports their true belief. □ □

This property is the mathematical foundation of NOTCH's integrity. A predictor who systematically overstates or understates their confidence will, in expectation, produce a worse Brier Score than one who reports honestly. No strategy of confidence manipulation improves the expected score. This holds regardless of the predictor's underlying accuracy—even a mediocre predictor benefits from honest confidence reporting.

## 3.2 The Notch Score

The raw Brier Score measures calibration but does not capture the full picture of a predictor's value. A predictor with five predictions and a perfect Brier Score is less trustworthy than one with 500 predictions and a strong (but imperfect) Brier Score. We define a composite metric.

**Definition 3.2** (Notch Score)**.** *The Notch Score for a predictor with $N$ resolved predictions is:*

$$\text{NS} = \alpha \cdot (1 - \text{BS}) + \beta \cdot A + \gamma \cdot V(N) + \delta \cdot C(t) \tag{7}$$

*where:*

- BS *is the time-weighted Brier Score (see below),*
- $A = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[o_i = \text{correct}_i]$ *is the raw accuracy rate,*
- $V(N) = \min\left(1, \frac{\log(1+N)}{\log(1+N_{\text{ref}})}\right)$ *is a volume component that saturates at a reference count $N_{\text{ref}}$ (initially 500),*
- $C(t) \in [0, 1]$ *is a consistency score based on the regularity of prediction activity over time,*
- $\alpha, \beta, \gamma, \delta \geq 0$ *are weights satisfying $\alpha + \beta + \gamma + \delta = 1$.*

**Initial weight configuration:** $\alpha = 0.40$, $\beta = 0.25$, $\gamma = 0.20$, $\delta = 0.15$. These weights are governance-adjustable. The heavy weighting toward calibration ($\alpha$) reflects the protocol's thesis that calibration quality—knowing *how much* you know—is more informative than raw hit rate.

### 3.3 Time-Weighted Decay

A predictor who was accurate two years ago but has been inactive for six months should not retain an unchanged score. We apply an exponential moving average (EMA) decay to the Brier Score calculation:

$$\text{BS}_{\text{ema}} = \frac{\sum_{i=1}^{N} \lambda^{t_{\text{now}} - t_i} (f_i - o_i)^2}{\sum_{i=1}^{N} \lambda^{t_{\text{now}} - t_i}} \tag{8}$$

where $\lambda \in (0, 1)$ is the decay factor (initially 0.95 per month, *i.e.* a half-life of approximately 13.5 months) and $t_i$ is the resolution timestamp of prediction $i$. Recent predictions receive more weight. This creates a "use it or lose it" dynamic: predictors must remain active to maintain their score.

### 3.4 Difficulty Adjustment

Not all predictions are equally informative. Predicting that Bitcoin will remain above $1,000 in 24 hours is trivially easy; predicting it will exceed $97,542 is substantially harder. The scoring system adjusts for difficulty.

**Definition 3.3** (Difficulty Multiplier)**.** *For a prediction on asset $a$ with target price $\tau$ and current price $p_0$ at commit time, with recent annualized volatility $\sigma_a$ and time horizon $\Delta t$ (in days):*

$$D = 1 + \kappa \cdot \frac{|\tau - p_0| / p_0}{\sigma_a \sqrt{\Delta t / 365}} \tag{9}$$

*where $\kappa > 0$ is a scaling constant (initially 2.0). The numerator is the relative distance to the target price; the denominator is the expected price movement at the given volatility and horizon. A correct prediction on a difficult call (high $D$) produces a larger positive score update; an incorrect prediction produces a proportionally larger negative update.*

The difficulty multiplier is recorded at commit time based on the oracle's reported price and volatility, ensuring that it cannot be manipulated after the fact.

### 3.5 Sybil Resistance

A natural attack vector is the "Sybil lottery": create $K$ wallets, submit random predictions from each, and promote whichever wallet achieves the highest accuracy by chance. We show this strategy fails against calibration scoring.

Consider a Sybil attacker who creates $K$ wallets, each making $N$ predictions at constant stated confidence $c$ on events with true base rate $p = 0.5$ (*i.e.*, coin-flip difficulty). The expected accuracy of the best wallet after $N$ predictions follows a well-known order-statistic result: for large $K$, the maximum accuracy converges to:

$$A_{\max}^{(K,N)} \approx \frac{1}{2} + \frac{1}{2}\sqrt{\frac{2 \ln K}{N}} \tag{10}$$

With $K = 100$ wallets and $N = 200$ predictions, the luckiest wallet achieves approximately $A_{\max} \approx 0.65$. However, this wallet has stated confidence $c$ on all predictions. Its Brier Score is:

$$\mathrm{BS}_{\mathrm{sybil}} \approx A_{\max}(c-1)^2 + (1-A_{\max})c^2 \tag{11}$$

If the attacker set $c = 0.95$ (as they would need to in order to appear highly confident), this yields $\mathrm{BS}_{\mathrm{sybil}} \approx 0.65 \cdot 0.0025 + 0.35 \cdot 0.9025 \approx 0.318$—a poor Brier Score that exposes the wallet as badly miscalibrated despite its above-average accuracy. A genuinely skilled predictor with the same 65% accuracy who honestly reports $c = 0.65$ achieves BS $\approx 0.228$—measurably and visibly better.

The Notch Score compounds this defense. The volume component $V(N)$ penalizes wallets with few predictions. The consistency component $C(t)$ penalizes wallets that appear suddenly and lack a history. The staking requirement imposes an economic cost on each wallet. Together, these make Sybil attacks both statistically detectable and economically expensive.

### 3.6 Calibration Visualization

The Notch Score includes a per-predictor calibration curve: a plot of stated confidence (x-axis) against observed outcome frequency (y-axis). A perfectly calibrated predictor falls on the diagonal: when they say "80% confident," they are correct 80% of the time. This curve is published on each predictor's profile and serves as an immediate visual indicator of reliability.

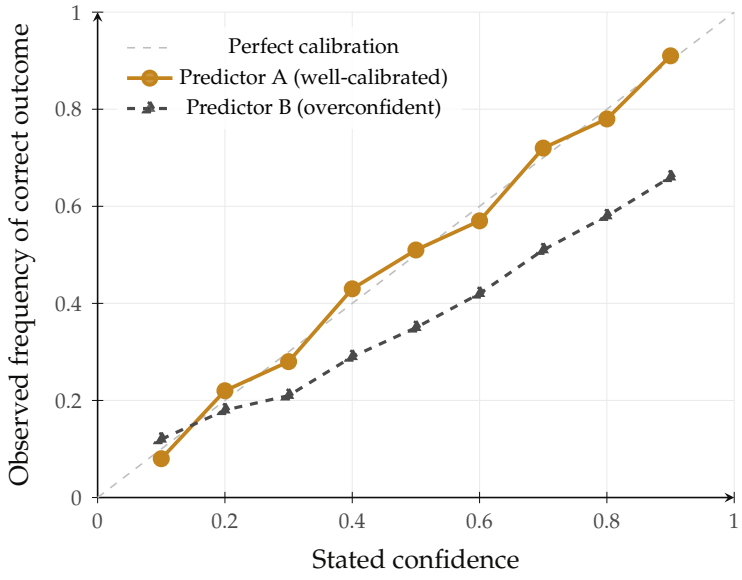


Figure 2: Calibration curves for two predictors. Predictor A (solid, amber) tracks the perfect-calibration diagonal closely. Predictor B (dashed, gray) systematically overstates confidence: when they say "90% confident," they are correct only 66% of the time. The Brier Score captures this divergence quantitatively.

# 4 Instrument Layer

The scoring mechanism produces a verifiable, on-chain reputation signal. The instrument layer transforms that signal into a set of tradeable financial products with continuous price discovery. Three instrument classes are defined.

## 4.1 Alpha Passes

An Alpha Pass is a tokenized right to receive a predictor's future predictions. It is the primary product in the NOTCH marketplace.

**Definition 4.1** (Alpha Pass)**.** *An Alpha Pass $\mathcal{P}(x, n)$ grants the holder access to the next $n$ predictions from predictor $x$, where $n \in \mathbb{N}^+$ is set by the predictor at issuance. The pass is implemented as an ERC-1155 token with the following* `tokenId` *encoding:*

$$\texttt{tokenId} = (\texttt{uint256}(x) \ll 128) \mid (\texttt{uint128}(\textit{epoch})) \tag{12}$$

*where $x$ is the predictor's address (truncated) and epoch is a sequential issuance counter.*

Passes are issued by predictors and sold on the primary market at a price determined by the predictor, subject to a floor derived from the predictor's Notch Score (preventing reputable predictors from pricing below their demonstrated value). Passes are freely transferable and tradeable on a secondary market operated by the protocol's exchange contract.

The protocol collects a fee of $\phi_p = 2.5\%$ on every primary and secondary market pass transaction.

Pass holders receive predictions via an encrypted channel. At reveal time, the prediction is published to pass holders through an access-gated API. After resolution, the prediction becomes public for scoring verification purposes. This creates a temporal information advantage: pass holders see predictions before the public, but the public can verify after the fact that the predictions existed (and were scored).

## 4.2 Alpha Futures

Alpha Futures are binary options on the trajectory of a predictor's Notch Score. They enable the market to price the *sustainability* of a predictor's edge—not merely whether a single prediction is correct, but whether a predictor's overall quality will increase or decrease over a defined period.

**Definition 4.2** (Alpha Future)**.** *An Alpha Future $\mathcal{F}(x, T, \theta)$ is a binary contract that pays 1 unit if predictor $x$'s Notch Score at time $T$ exceeds threshold $\theta$, and 0 otherwise:*

$$\textit{Payoff}(\mathcal{F}) = \begin{cases} 1 & \textit{if } \mathrm{NS}(x, T) > \theta \\ 0 & \textit{otherwise} \end{cases} \tag{13}$$

*Resolution is deterministic: the on-chain Notch Score at time $T$ is queried, and the contract settles automatically. No oracle is required beyond the score itself.*

Alpha Futures are traded on a Uniswap V4 Hook pool [9], with dynamic fees adjusted by the hook contract based on instrument volatility and liquidity depth. The hook implements automated time-weighted average market maker (TWAMM) execution for protocol buyback operations.

### 4.3 Alpha Indices

Alpha Indices provide diversified exposure to the aggregate prediction quality of a ranked cohort of predictors.

**Definition 4.3** (Alpha Index)**.** *An Alpha Index $\mathcal{I}_k$ bundles the top $k$ predictors by Notch Score, weighted proportionally:*

$$w_j = \frac{\mathrm{NS}_j}{\sum_{i=1}^{k} \mathrm{NS}_i}, \quad j = 1, \ldots, k \tag{14}$$

*The index rebalances at fixed epochs (initially monthly). A holder of one index share is entitled to receive the weighted average pass output of all constituent predictors.*

The index is the protocol's solution to the liquidity fragmentation problem. Individual Alpha Futures for low-ranked predictors may have insufficient trading interest to support efficient price discovery. The index aggregates demand: a consumer who wants broad exposure to verified prediction quality need not evaluate individual predictors. This mirrors the relationship between individual stock analysis and S&P 500 index investing.

The protocol charges a management fee of $\phi_m = 0.5\%$ annually on capital following index signals.

### 4.4 System Architecture

Figure 3 shows the contract architecture and data flow between the five core contracts.

### 4.5 Cross-Protocol Composability

The Notch Score is designed as a composable on-chain primitive. Any smart contract on the deployment chain can query a predictor's score through a minimal interface:

```
interface INotchScore {
    function getScore(address predictor)
        external view returns (uint256);
    function getCalibration(address predictor)
        external view returns (uint256);
    function getPredictionCount(address predictor)
        external view returns (uint256);
    function isActive(address predictor)
        external view returns (bool);
}
```

This enables external applications: lending protocols that offer preferential rates to well-calibrated predictors, trading platforms that surface Notch Scores on user profiles, or governance systems that weight votes by demonstrated forecasting ability. Each integration extends the protocol's utility without requiring changes to the core contracts.

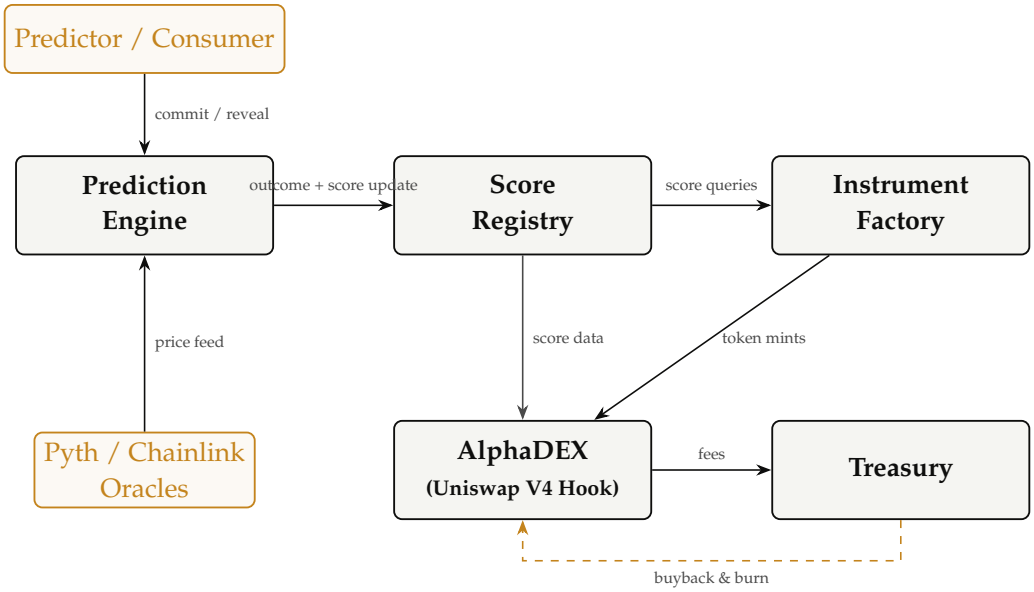


Figure 3: System architecture. Five core contracts interact through defined interfaces. The `PredictionEngine` manages the commit-reveal lifecycle, the `ScoreRegistry` maintains Notch Scores, the `InstrumentFactory` mints ERC-1155 tokens, the `AlphaDEX` (a Uniswap V4 Hook) facilitates trading, and the `Treasury` manages fee collection and automated buybacks.

# 5 Oracle Design and Security

The integrity of the scoring system depends entirely on the integrity of the price data used for resolution. A compromised oracle can make incorrect predictions appear correct, undermining the entire reputation layer. We adopt a defense-in-depth approach.

## 5.1 Dual Oracle Strategy

The protocol employs two independent price oracle networks:

- **Primary: Pyth Network.** A pull-based oracle with sub-second update latency and pay-per-use pricing. The resolver submits a Pyth price attestation along with the `resolve()` call, and the contract verifies the attestation on-chain. Cost per resolution: approximately $0.001.
- **Fallback: Chainlink Data Feeds.** A push-based oracle with broader asset coverage and established reliability in DeFi. Used when Pyth attestations are unavailable or when the resolution is disputed.

At resolution time, the contract queries the primary oracle. If the primary oracle price and the fallback oracle price diverge by more than a configurable threshold $\epsilon$ (initially 0.5%), the resolution is paused and enters a dispute window.

## 5.2 TWAP Smoothing

To prevent single-block price manipulation, the settlement price $p^*$ is computed as a time-weighted average price (TWAP) over a short window surrounding the expiry timestamp:

$$p^* = \frac{1}{W} \int_{t_{\text{exp}}-W/2}^{t_{\text{exp}}+W/2} p(t)\, dt \tag{15}$$

where $W$ is the smoothing window (initially 5 minutes). This ensures that a flash loan or atomic transaction that momentarily distorts the price at the exact expiry timestamp does not affect the resolution.

### 5.3 Dispute Mechanism

If an oracle reading appears anomalous, any participant may stake $NOTCH tokens to open a dispute within a 24-hour window after resolution. The dispute triggers a secondary resolution using a broader set of data sources (multiple Chainlink nodes, exchange API price snapshots recorded by a decentralized keeper network). If the dispute is upheld, the original resolution is overturned, the disputed prediction is re-scored, and the disputer receives a reward from the staked collateral of the original resolver. If the dispute is rejected, the disputer forfeits their stake.

This mechanism is inspired by UMA's Optimistic Oracle [10], which has demonstrated reliability at scale in Polymarket with over 7,000 monthly proposals.

### 5.4 Circuit Breakers

The protocol implements automatic circuit breakers for extreme conditions:

- **Price deviation halt:** If the oracle-reported price deviates by more than $3\sigma$ from the 24-hour moving average, resolutions for that asset are paused for one hour.
- **Oracle staleness check:** If the most recent oracle update is older than a configurable threshold (initially 60 seconds for Pyth, 3,600 seconds for Chainlink), the resolution is deferred.
- **Global pause:** A governance-controlled emergency pause can halt all resolutions. The pause is subject to a 48-hour timelock for activation and a 7-day timelock for removal, preventing both rash decisions and permanent censorship.

## 6 Token Economics

The $NOTCH token serves three functions within the protocol: staking collateral for prediction commitment, governance voting, and fee medium. The token is designed to accrue value from protocol activity through a buyback-and-burn mechanism.

### 6.1 Supply and Distribution

Total supply is fixed at 1,000,000,000 $NOTCH and is immutable—no minting function exists in the token contract.

### 6.2 Staking Mechanism

Predictors must stake a minimum of $S_{\text{min}}$ $NOTCH tokens to commit predictions. The staking requirement creates persistent demand for the token proportional to the active predictor base. With $M$ active predictors each staking $S_{\text{min}}$ tokens:

| Allocation | Vesting | Share |
|---|---|---|
| Community pool | 4-year linear | 40% |
| Predictor rewards | Earned by score | 15% |
| Early user airdrop | At TGE | 10% |
| Ecosystem grants | Governance-approved | 10% |
| Liquidity incentives | 2-year decay | 5% |
| Protocol treasury | Governance-controlled | 20% |
| Founding team | 1-year cliff, 3-year vest | 15% |
| Initial DEX liquidity | Concentrated LP position | 15% |

Table 1: Token allocation. No allocation exceeds 40% to any single category. Team tokens are fully locked for 12 months.

$$\text{Locked supply from staking} = M \cdot S_{\min} \tag{16}$$

Staked tokens are subject to slashing in two cases: (i) a predictor commits a prediction and fails to reveal within the allowed window, or (ii) provably manipulative behavior is demonstrated through a governance dispute. Slashed tokens are transferred to the protocol treasury. The expected slashing rate under normal operation is 1–2% of staked tokens per year.

### 6.3 Revenue Flow and Buyback-Burn

All protocol revenue flows to the `Treasury` contract. Revenue sources include Alpha Pass trading fees ($\phi_p = 2.5\%$), Alpha Future and Index trading fees, API subscription fees, and Alpha Index management fees ($\phi_m = 0.5\%$). The treasury executes an automated allocation:

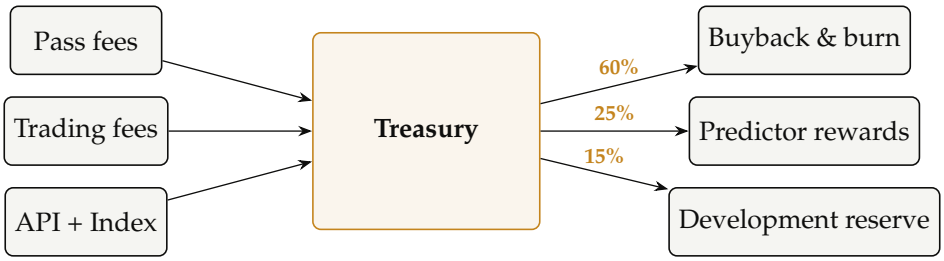


Figure 4: Revenue flow. All protocol fees pass through the Treasury contract. 60% is used for automated TWAMM buyback-and-burn of $NOTCH, permanently reducing supply. 25% is distributed to high-scoring predictors. 15% is reserved for development.

The 60% buyback-and-burn allocation is executed via time-weighted average market maker (TWAMM) orders on the Uniswap V4 pool, spreading buy pressure across time to minimize market impact. Purchased tokens are sent to the zero address and are permanently removed from circulation.

### 6.4 Deflationary Dynamics

After the 4-year community pool vesting period concludes, no new tokens enter circulation. From that point forward, total supply can only decrease through burns. The rate of supply reduction is a function of protocol revenue:

$$\frac{d(\text{Supply})}{dt} = -0.60 \cdot \frac{R(t)}{P(t)} \tag{17}$$

where $R(t)$ is instantaneous protocol revenue and $P(t)$ is the token price. Higher revenue and lower token prices accelerate the burn; higher token prices decelerate it. This creates a natural price-stabilizing feedback loop: if the token price drops sharply but revenue remains constant, more tokens are bought and burned per unit of revenue, contracting supply and supporting price recovery.

### 6.5 Governance

Token holders vote on protocol parameters including fee rates ($\phi_p$, $\phi_m$), scoring weights ($\alpha, \beta, \gamma, \delta$), staking minimums ($S_{\min}$), oracle configurations, and treasury allocation percentages. Governance uses a time-locked voting mechanism: proposals require a 3-day voting period and a 2-day execution delay, during which the community can review and, if necessary, veto through a higher-threshold counter-vote.

## 7 Implementation and Deployment

### 7.1 Deployment Target

Arbitrum One is the primary deployment chain. Post-Dencun upgrade, Arbitrum transaction costs average \$0.005–\$0.03, making the complete commit-reveal-resolve cycle economically viable at approximately \$0.01–\$0.03 per prediction. Arbitrum's full EVM equivalence, large DeFi ecosystem (including Uniswap V3/V4, Chainlink feeds, and established user base), and Stylus support for Rust-compiled contracts provide a mature foundation.

Base serves as a secondary deployment for user acquisition through Coinbase's 110M+ verified user base. HyperEVM, with its native order book precompiles and high throughput, is a V2 target once its ecosystem matures.

### 7.2 Contract Architecture

The protocol comprises five interdependent contracts, as depicted in Figure 3. All contracts are written in Solidity 0.8.24+ and developed using the Foundry framework. The `Prediction` struct is storage-optimized to fit within two 256-bit slots:

```
struct Prediction {
    bytes32 commitHash;       // slot 1: 32 bytes
    uint64  commitTimestamp;  // slot 2: 8 bytes (packed)
    uint64  expiryTimestamp;  //         8 bytes (packed)
    uint64  revealedAt;       //         8 bytes (packed)
    uint8   status;           //         1 byte  (packed)
}
```

```
// Total: 2 storage slots (64 bytes)
```

Gas costs per operation:

- `commit()`: ∼45,000 gas ($0.004 on Arbitrum)
- `reveal()`: ∼65,000 gas ($0.006 on Arbitrum)
- `resolve()`: ∼80,000 gas ($0.008 on Arbitrum), including oracle query

For high-frequency predictors, a Merkle batch commitment mode allows $n$ predictions to be committed under a single root hash, reducing the per-prediction commit cost by approximately 90% for batches of 10 or more.

## 7.3 Security

The protocol's security strategy consists of four layers:

1. **Formal verification** of critical scoring and commitment logic using Certora or Halmos.
2. **Competitive audit** via Code4rena or Sherlock, with a bounty pool of $60,000–$80,000.
3. **Boutique firm review** of oracle integration and economic invariants (Spearbit or Trail of Bits), budgeted at $30,000–$50,000.
4. **Bug bounty program** on Immunefi with a $50,000–$100,000 reward pool, active from testnet onward.

Total security budget: $140,000–$235,000. This is a non-negotiable commitment. A protocol that handles financial instruments derived from reputation cannot afford a contract vulnerability.

## 7.4 Timeline

| Phase | Duration | Deliverables |
|---|---|---|
| Core contracts | Months 1–5 | PredictionEngine, ScoreRegistry, oracle integration |
| Instrument layer | Months 4–8 | InstrumentFactory, AlphaDEX (V4 Hook), Treasury |
| Frontend & API | Months 6–10 | Dashboard, API, Telegram bot |
| Audit & testnet | Months 8–12 | Formal verification, competitive audit, public testnet |
| Mainnet | Months 12–14 | Arbitrum One deployment, Genesis Season |
| Token generation | Months 14–16 | $NOTCH launch, airdrop, DEX liquidity |

Table 2: Development timeline. Phases overlap where dependencies permit. Token generation occurs only after the protocol has demonstrated product-market fit on mainnet.

## References

[1] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, 1950.

[2] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[3] A. H. Murphy, "A new vector partition of the probability score," *Journal of Applied Meteorology*, vol. 12, no. 4, pp. 595–600, 1973.

[4] L. J. Savage, "Elicitation of personal probabilities and expectations," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 783–801, 1971.

[5] Polymarket, "Polymarket documentation," https://docs.polymarket.com, 2024.

[6] Kalshi, "Kalshi exchange: CFTC-regulated event contracts," https://kalshi.com, 2025.

[7] Copin.io, "On-chain trader analytics and copy-trading," https://copin.io, 2025.

[8] Numerai, "Numerai tournament: Machine intelligence for hedge funds," https://numer.ai, 2023.

[9] H. Adams, M. Salem, and S. Catania, "Uniswap v4 Core," Uniswap Labs, 2024.

[10] UMA Protocol, "Optimistic Oracle v3: Documentation," https://docs.uma.xyz, 2024.

[11] Chainlink Labs, "Chainlink Data Feeds: Technical documentation," https://docs.chain.link/data-feeds, 2024.

[12] Pyth Network, "Pyth Price Feeds: Pull oracle architecture," https://docs.pyth.network, 2024.

[13] N. Johnson, "EIP-137: Ethereum Name Service specification," Ethereum Improvement Proposals, 2017.