
Problem setup (notation)

We operate on discrete time windows indexed by t (e.g., 1-hour windows) and spatial units indexed by g (grid cells / block groups). For a given (g,t) we want a consensus probability that a murder event occurs in that cell during time window t.

Let:

- N = number of independent oracle operators (universities, nonprofits, municipal nodes).
- For operator $i\in\{1,\dots,N\}$ at time t, operator i ingests public inputs $X_{i,g,t}$ (license-plate reader counts, people counter counts, other covariates restricted to public/aggregated data).
- Operator i runs the published model container and outputs a probabilistic forecast $p_{i,g,t}\in[0,1]$ = estimated probability of at least one murder in (g,t).
- Operator i also outputs a signed attestation $A_{i,g,t}$ consisting of:
    - $D_{i,g,t}$ = SHA-256 digest of the raw input snapshot they used,
    - M_{v} = model container version hash,
    - $p_{i,g,t}$ (or a hash $H_{i,g,t}=\text{SHA256}(p_{i,g,t}\,\|\,\text{meta})$ in commit stage),
    - timestamp $T_{i,g,t}$,
    - digital signature $\sigma_i(\cdot)$.
- The smart contract stores only digests / signatures and minimal metadata; raw inputs remain off-chain.

We maintain a **reputation weight** $w_{i,t}\ge 0$ for each operator i that evolves over time according to predictive performance.

Core aggregation math

1) Robust ensemble probability

A straightforward robust aggregation is the *weighted trimmed mean* or *Huber* style estimator, but a simple and mathematically clean choice is:

Weighted average:

$\bar p_{g,t} = \frac{\sum_{i=1}^N w_{i,t}\, p_{i,g,t}}{\sum_{i=1}^N w_{i,t}}$.

Ensemble uncertainty (variance):

$\sigma^2_{g,t} = \frac{\sum_{i=1}^N w_{i,t}\, (p_{i,g,t} - \bar p_{g,t})^2}{\sum_{i=1}^N w_{i,t}}$.

If you want heavy outlier resistance, replace the weighted mean with a weighted **median** or **trimmed mean**: sort $p_{i,g,t}$ by value, drop the top/bottom $\alpha\%$ by weight, then average remaining.

2) Consensus acceptance test

Because operators may differ slightly, we need a rule for when the set of attestations is considered to have reached *consensus*:

Option A — **Hash agreement (strict)**: If at least K operators (e.g., $K=\lceil 2N/3\rceil$) submit identical hashes $H_{i,g,t}$ (i.e., they got the same digest for inputs and the same deterministic model produced identical forecast), accept that canonical forecast. This is appropriate when all operators use deterministic code and identical inputs.

Option B — **Probabilistic consensus (recommended for noisy inputs)**: Accept consensus if:

1. the ensemble probability $\bar p_{g,t}$ is above a decision threshold τ (e.g., 0.002 for very rare events like murder), **and**

2. the relative dispersion is low: $\dfrac{\sigma_{g,t}}{\bar p_{g,t} + \epsilon} \le \delta$ (e.g., $\delta=1.0$), meaning operators broadly agree proportionally, **or**

3. there exists a super-majority K of operators whose individual probabilities lie within a confidence neighborhood of the ensemble: $|p_{i,g,t}-\bar p_{g,t}|\le \gamma$ for at least K operators. Typical γ could be $0.5\times\bar p$ or absolute 0.01 depending on base rates.

The smart contract records the ensemble $\bar p_{g,t}$, $\sigma_{g,t}$, the set of is that attested, and the input/model digests.

Reputation (weight) update — performance-based

We maintain weights $w_{i,t}$ reflecting historical calibration/accuracy. Use an *exponentiated gradient* style update based on a proper scoring rule (Brier or log loss).

Define actual outcome for (g,t): $y_{g,t}\in\{0,1\}$ observed after window ends.

Use Brier score (squared error):

$$\ell_{i,g,t} = (p_{i,g,t} - y_{g,t})^2.$$

Update rule (multiplicative / exponential weighting):

$$\tilde w_{i,t+1} = w_{i,t} \cdot \exp\big(-\eta \,\ell_{i,g,t}\big),$$

then normalize:

$$w_{i,t+1} = \frac{\tilde w_{i,t+1}}{\sum_{j=1}^N \tilde w_{j,t+1}} \cdot W_{\text{scale}}$$

where $\eta>0$ is a learning rate (e.g., $\eta=0.5$), and W_{scale} preserves an absolute scale if desired (or keep normalized weights summing to 1).

This means operators who predict closer to truth gain relative influence; poor performers shrink. If events are extremely rare (sparse $y=1$), you may want to use log loss with smoothing or pool updates over multiple cells to avoid noisy weight swings.

Alternatively, track *calibration* and *reliability* separately:

- Calibration error: $c_{i} = \mathbb{E}[(p_{i} - y)]$ estimated over a sliding window.
- Brier skill for ranking.

Combine into composite reputation score.

Collusion and anomaly detection

We need mechanisms to detect collusion or compromised nodes.

1) Pairwise correlation test

Compute pairwise Pearson correlation r_{ij} between prediction sequences $\{p_{i,g,t}\}_t$ and $\{p_{j,g,t}\}_t$ over a window. High r_{ij} across many pairs at near-identical values is suspicious if their inputs should differ.

2) Consensus entropy / Simpson index

Define the effective number of distinct predictions (diversity):

$$\mathrm{EffCount} = \exp\left(-\sum_{k} \tilde q_k \log \tilde q_k\right)$$

where $\tilde q_k$ are weights for unique prediction clusters. Low EffCount indicates low diversity — investigate.

3) Outlier detection

If an operator's forecast $p_{i,g,t}$ satisfies

$$\frac{|p_{i,g,t}-\bar p_{g,t}|}{\sqrt{\sigma^2_{g,t} + \epsilon}} > z_{\text{thresh}}$$

(e.g., $z_{\text{thresh}}=3$), mark as outlier. Log and optionally exclude from consensus for that round.

Operators flagged repeatedly are referred to the accountability board.

Commit-reveal for anti-front-running & tamper evidence

To avoid operators changing outputs after seeing others' submissions, use a two-step commit-reveal:

1. **Commit stage** (off-chain or on-chain short record): each operator i submits $C_{i} = \text{SHA256}(D_{i}\,\|\,M_v\,\|\,H_{i}\,\|\,T_{i})$ to the contract. This is cheap and does not reveal p_{i}.

2. **Reveal stage** (within a fixed window): reveal the tuple $(D_i, M_v, p_{i}, \sigma_i)$. The contract checks SHA256 matches C_i and signatures are valid. If a node fails to reveal in window, its commit is void for that round.

This provides non-repudiable evidence of what each operator used and produced.

Discrete hotspot consensus (top-K cells)

If you want consensus on *which cells are hotspots* (top-K highest risk), proceed:

- Each operator submits an ordered list $S_{i,t} = \{g_{i,1},\dots,g_{i,K}\}$ of top K cells by its scores.

- Convert each list into a score vector: for operator i, cell g gets score $s_{i,g} = \frac{K-r}{K}$ where r is rank (or zero if not present).

- Compute aggregated score:

$$S_g = \sum_{i=1}^N w_{i,t}\, s_{i,g}.$$

- The consensus top-K are the K cells with highest S_g. The smart contract can record these with their scores and operator attestations.

Jaccard overlap and rank correlation (Kendall Tau) between operators and consensus can be used as sanity checks.

Decision policy thresholding

Actionable decisions should be human-in-the-loop. A simple decision rule:

Flag cell g if:

$$\bar p_{g,t} \ge \tau \quad\text{and}\quad \frac{\sigma_{g,t}}{\bar p_{g,t} + \epsilon} \le \delta,$$

and at least K operators’ commits were valid and revealed. Choose τ based on historical precision targets (e.g., set τ so that top X% of cells contain Y% of historical murders). Because murder is rare, set thresholds conservatively and require human review.

Auditability & evidence published on-chain

For each accepted consensus round the following minimal metadata is written on-chain:

- Round ID, t.
- Ensemble probability $\bar p_{g,t}$ (or quantized to preserve some privacy).
- Ensemble variance $\sigma^2_{g,t}$.
- List of operator IDs and their commit hashes C_i.
- Model version hash M_v.
- Input dataset digests (canonical set) $\{D_j\}$.
- Link to public dashboard (off-chain full logs).

Because raw PII is never on-chain, auditors must fetch raw snapshots from the public portal and verify SHA digests match the recorded D_j.

Example (toy numeric)

Suppose $N=5$ operators; initial equal weights $w_i=1$. Their predictions for cell g at t:

$p=[0.004,\,0.002,\,0.005,\,0.003,\,0.90]$ — note one operator (5) gives a very large outlier, perhaps due to misconfigured input.

Weighted mean:

p¯=15∑pi=0.004+0.002+0.005+0.003+0.905=0.1828.\bar p = \frac{1}{5}\sum p_i = \frac{0.004+0.002+0.005+0.003+0.90}{5}=0.1828.

Variance dominated by outlier; robust approach: exclude any pip_i with z-score > 3:

Compute mean w/o operator5: p¯−5=(0.004+0.002+0.005+0.003)/4=0.0035\bar p_{-5}= (0.004+0.002+0.005+0.003)/4 = 0.0035. Relative dispersion shows operator5 is an extreme outlier (|0.90−0.0035| ≫ threshold). Protocol flags operator5, excludes it for consensus, and records the anomaly.

Consensus then uses p¯=0.0035\bar p=0.0035. With threshold τ=0.01\tau=0.01, not flagged for action.

This demonstrates robustness: one compromised/misconfigured oracle cannot force false alarm.

Pseudocode (high level)

```
Inputs: N operators, K_required, threshold tau, dispersion delta, z_thresh

For each round (g,t):
  1. Commit phase:
     For each operator i:
       D_i = hash(raw_input_snapshot)
       M_v = model_version_hash
       p_i = run_model(D_i, M_v)
       H_i = hash(p_i || meta)
       C_i = hash(D_i || M_v || H_i || timestamp)
       submit_commit(i, C_i)

  2. Reveal phase:
     For each operator i:
       reveal(i, D_i, M_v, p_i, signature)
       verify C_i == hash(D_i || M_v || hash(p_i || meta))
       verify signature

  3. Aggregation:
     Collect valid reveals V = {i: verified}
     Compute weighted mean p_bar and variance sigma^2 (using weights w_i)
     For each i in V compute z_i = |p_i - p_bar| / sqrt(sigma^2 + eps)
```

```
Exclude O = {i: z_i > z_thresh}  # outliers
Recompute p_bar using V \ O
Compute relative_dispersion = sigma / (p_bar + eps)

4. Acceptance:
  If (p_bar >= tau) AND (relative_dispersion <= delta) AND (|V \ O| >= K_required):
     Record consensus on-chain: store p_bar, sigma, list of commits, model version
  Else:
     Record non-consensus or low confidence

5. Weight update (after observing outcome y):
  For each i in V:
    loss = (p_i - y)^2
    w_i <- w_i * exp(-eta * loss)
  Normalize weights
```

Practical considerations

- **Extremely rare events**: murder is rare; forecasts are low-probability. Use large spatial aggregation when training to get statistical power; aggregate small cells into microregions for modeling and apply downscaling only when warranted.

- **Pooling evidence across covariates**: license-plate spikes and people-counts should be standardized (z-scores) relative to historical baselines to avoid raw count scale differences between sensors.

- **Delay in ground truth**: murders may be reported with delay; use a time window and lock epoch for outcome labeling.

- **Operator diversity**: ensure operators fetch inputs from independent endpoints where possible (e.g., different mirrors) to reduce correlated failures.

- **Governance**: the accountability board handles repeated anomalies, requests reprocessing, and can require re-run when new evidence appears.
