The Verifiable Quant Layer

On What On-Chain Protocols Must Import,
and the Trust Each Import Requires

Mehdi Rhouzlane

July 2026

Abstract

An on-chain protocol makes quantitative decisions — a loan-to-value ratio, a liquidation price, a margin requirement — but it cannot compute them. The Ethereum Virtual Machine cannot run the volatility, correlation, and liquidity estimates that mature finance uses to price risk, and even where it could, the inputs are exogenous and adversarially supplied. Every protocol therefore already *imports* its intelligence — from governance votes, from risk curators, from oracles — but does so without a theory of trust. That gap, not exotic cleverness, is what the large credit losses of this cycle actually were: a high-value decision riding on a cheaply-corrupted input. I name the object being imported — the map from market state to risk decision, the *quant layer* — and give a taxonomy of the trust regimes under which it can cross the chain boundary, from discretionary governance to cryptographic proof. These regimes trace a frontier: sophistication, trustlessness, and low latency cannot be had at once. The design rule that follows is sharp, and today absent — assign each decision to the cheapest regime whose *cost-to-corrupt* exceeds the value a corrupted decision would leak.

1 The heuristic ceiling

Open the risk configuration of any on-chain lending market and you find constants. A collateral factor of 0.8. A liquidation threshold of 0.85. An interest rate that is a piecewise-linear function of utilization with a kink set by a vote. An oracle whose manipulation resistance is a time-averaging window chosen by feel. Each is a number a human picked and a human revises. In an options clearing house none of them would be a constant; each would be the output of a model fed by live volatility, correlation, and liquidity, revised continuously as the market moves.

Two facts keep the constants in place. The first is mechanical: the virtual machine cannot run the models. Computation on-chain is metered and capped by the block gas limit, there is no native floating point, and long price histories cannot be held or replayed cheaply, so a covariance estimate, an implied-volatility surface, or a market-impact model is simply infeasible to evaluate in a transaction. This is not a contested claim; it is the standard justification for oracles in the first place, which exist precisely to perform a computation that "may have been infeasible to calculate on-chain" [1].

The second fact is deeper, and survives even if the first is someday engineered away. A protocol that *could* run the model still could not trust its inputs. Volatility, correlation, order-book depth, the price itself — all are exogenous to the chain, produced somewhere else, and delivered by a party who may profit from lying. A model is only ever as trustworthy as the state it is fed.

So every protocol already outsources its intelligence. It is the one question the design never asks out loud: *under what trust regime?* The claim of this note is that the answer is usually implicit, usually the same for every decision regardless of stakes, and that this is exactly why things break. The large credit failures of this cycle were not clever exploits of sound systems.

They were the predictable cost of importing a high-stakes computation under a trust regime too cheap to corrupt.

Fix the object first. A protocol decision is a map

$$D = f(\theta), \tag{1}$$

where f is a policy — the LTV rule, the liquidation trigger, the margin formula — and θ is the market state it consumes: prices, volatilities, correlations, liquidity. The chain is very good at one half of this. It can *enforce* f cheaply and trustlessly: hold the collateral, run the arithmetic of the rule, move the funds if the condition trips. What it cannot do is produce a trustworthy θ, or evaluate a sophisticated f, without importing it from outside. The scarce and dangerous component is therefore a **trustworthy** (f, θ) **crossing the boundary**. Call whatever supplies that crossing the *quant layer*, and the design question becomes: how do you verify it?

2 The intelligence a protocol needs — and cannot produce

Four computations do most of the work in any serious risk engine, and each fails on-chain for the same three reasons at once — it is compute-heavy, hungry for off-chain data, and manipulable exactly where it enters.

- *Volatility and covariance.* Sizing a haircut needs an estimate of how violently collateral moves and how it co-moves with the debt: history plus floating-point linear algebra, unavailable on-chain and drawn from series a manipulator can bend.
- *Portfolio and tail risk.* A protocol's book is a portfolio, but its parameters are set per asset, as if correlations were zero. Real economic capital is a tail measure over the joint distribution, and it is not computable one silo at a time.
- *Liquidity and market impact.* The safe size of a liquidation depends on the depth available to absorb it; dumping collateral at an oracle mid-price assumes a liquidity that is not there. That depth lives in off-chain order books and obeys a square-root law of impact the chain cannot see [12].
- *Forward-looking calibration.* The information that matters is implied, not realized: the option surface prices the market's view of future risk, quoted on venues the protocol has no native access to.

The pattern is total. Everything required to price risk *well* is heavy, external, and corruptible at the input; everything the chain does natively — enforce the rule, move the asset — is light, internal, and trustless. Finance's intelligence and crypto's trust sit on opposite sides of a single boundary. The interesting engineering is the boundary itself.

3 Five ways to cross the boundary

There is no single way to bring (f, θ) on-chain; there are several, differing in the one variable that matters — **what, exactly, gets verified** — and therefore in cost, latency, and failure mode. From weakest guarantee to strongest:

R0 — Discretionary. Humans compute $f(\theta)$ off-chain and push the result through a privileged transaction. This is the status quo for risk parameters: a firm such as Gauntlet or Chaos Labs simulates the book and recommends collateral factors, caps, and curves, which a DAO enacts by vote [11]. Nothing is verified — you trust the actor's competence and honesty. Sophistication is unbounded; latency is days; the failure mode is staleness and capture.

R1 — Attested. A named party signs (f, θ) off-chain and the chain checks a signature. Pyth works this way: first-party publishers submit their own prices, an aggregate is formed, and

the consumer pulls a signed update into its own transaction, where the contract verifies the signatures before use — a feed that refreshes every 400 ms [2]. RedStone and Chronicle place signed payloads in calldata and verify them on demand; zkTLS notarizes that a datum came from a given web source. Verification is a signature check — fast and cheap — and sophistication is unbounded. You trust that the signer does not lie and is not compromised.

R2 — Aggregated. A committee computes independently and a threshold signature attests the aggregate. Chainlink's off-chain reporting has the nodes agree a single report, a quorum sign it, and one transaction carry the median on-chain, so cost is amortized across the committee [3]. You no longer trust one signer, but an honest majority; the failure mode is collusion, or a data source all of them share.

R3 — Optimistic. A value is *asserted* with a bond, a challenge window opens, and anyone may dispute by posting their own bond; undisputed assertions settle as true, and disputes escalate to a staked-token vote. This is UMA's optimistic oracle, and its trust is purely economic — the bond makes honesty the profitable strategy, provided at least one rational watcher exists [4]. Any assertion is expressible and the happy path is cheap; you pay in latency (the challenge window) and in the assumption that someone is watching.

R4 — Proven. The computation itself is proven correct. A zero-knowledge machine-learning system such as EZKL compiles a model into a circuit and emits a validity proof, checkable by an on-chain verifier, that the output really is f applied to the committed inputs [5]. Here you trust mathematics rather than any party. The price is severe: proving cost climbs with model size, inputs must be quantized, and anything approaching a large model is infeasible to prove on-chain today — so R4 buys its trustlessness by capping how sophisticated f can be.

Regime	**You trust**	**Verified by**	**Latency**	**Fails when**
R0 Discretionary	an actor	nothing	days	stale / captured
R1 Attested	a signer	a signature	sub-second	signer lies
R2 Aggregated	a majority	a threshold sig.	seconds–min	majority colludes
R3 Optimistic	a watcher exists	silence	challenge window	no one watches
R4 Proven	mathematics	a proof	proving time	bad input / bug

Table 1: The five regimes for importing (f, θ), ordered by strength of guarantee. Sophistication runs the other way: R0 and R1 impose no limit on the model, R4 the tightest.

One subtlety dissolves a common hope. Only R4 verifies the *computation*; R1, R2, and R3 verify a proxy for it — a signer, a majority, a silence. And even R4 does not verify that θ was *true*: a proof establishes that f was correctly applied to the input you were handed, not that the input described the world. The oracle problem — *is the input real?* — is never dissolved by cryptography. It is only ever pushed to the edge, where it becomes the irreducible question of who is trusted to sign the world.

4 The trilemma, and the one rule that follows

Lay the regimes against three properties a designer wants — the **sophistication** of the model that crosses, the **trustlessness** of the crossing, and its **latency and cost** — and no regime holds all three. R4 buys trustlessness by surrendering sophistication and speed. R1 buys sophistication and speed by surrendering trustlessness. R3 buys sophistication cheaply by surrendering latency. R0 buys sophistication alone and pays with everything else. None dominates; each is a corner of

a frontier. It follows that the perennial question — *which oracle, which risk system, is best?* — is a category error. There is no best; there is only a match between a decision and a regime.

What supplies the match is that a decision and a regime can be measured in the same unit: money. Every decision has an **exploitable value-at-risk** V_D — the profit an attacker extracts by corrupting it once. Every regime has a **cost-to-corrupt** C_R — the price of bribing the signer, buying the majority, out-bonding the disputers, or moving the manipulable input beneath it. A decision is safe under a regime precisely when corrupting it costs more than it pays:

$$\text{assign } D \text{ to the cheapest } R \text{ such that} \quad C_R > V_D. \tag{2}$$

That is the whole design rule, and it is almost never followed. Protocols route every decision — the slow parameter on a deep, hard-to-move asset and the live liquidation price on a thin, trivially-moved one — through the *same* regime, usually one oracle and one governance process. The rule says this is wrong in both directions at once: it over-pays for security on the decisions that never needed it and, fatally, under-secures the decisions where V_D is large and the input beneath the regime is cheap to move.

5 The ledger: mismatch versus bug

Read the incident record through the rule and it sorts into two piles. Either a loss was a *regime mismatch* — a decision whose cost-to-corrupt fell below its exploitable value — or it was something the quant layer was never going to fix.

Mango Markets is the pure case. In October 2022 an attacker used about $5M to inflate a thinly-traded perpetual roughly tenfold in an hour, drove the oracle that marked it as collateral, and borrowed about $110M against the inflated mark (∼$116M total per the SEC, ∼$67M later returned) [6]. A decision with an enormous V_D — mark-to-borrow — rode on an input whose C_R was $5M in a market no one else defended. The rule states the outcome in advance: that mark needed a regime whose cost-to-corrupt exceeded the borrow it enabled, or a liquidity-conditioned floor on how far it could move; it had neither. The shape recurs: **bZx** (2020), **Harvest** (2020), **Cheese Bank** (2020), and **PancakeBunny** (2021) all read the spot price of a shallow pool as truth, and all were corrected by a flash loan cheaper than the reserves it unlocked [8, 10].

A different mismatch is **latency**, not price. When an attacker built an outsized illiquid position against Aave in November 2022, the collateral could not be sold to cover the debt when the trade unwound, leaving ∼$1.6M of bad debt [9]. No oracle was fooled; the parameter simply lived in R0, whose latency is days — far too slow to reprice a concentration that moves in minutes. Tellingly, this is the migration now underway: firms like Chaos Labs increasingly push updates through automated *risk oracles* acting within governance-set bounds, dragging that decision out of R0 toward a faster regime [11].

Then the honest column. **Euler Finance** lost ∼$197M in March 2023 — the largest of them — and it was *not* a mismatch [7]. A donation function omitted a solvency check, letting the attacker force his own account into a bad state and self-liquidate at a discount; the imported θ was fine, and the fault lay in the on-chain f. A verified quant layer, however sophisticated, would not have prevented it. (The funds were later returned in full.) It has to be said plainly, because the value of the framework is its ability to *exclude*: separating the regime mismatches, which the taxonomy fixes, from the code bugs, which it does not, is itself the discipline. When you do the sort, most large credit losses fall in the first pile — the predictable consequence of importing a high-stakes computation under a regime too cheap to corrupt.

6 What protocols actually need

The lesson is not "a better oracle," and not "more decentralization" chanted at every layer. It is a **typed boundary**: each risk decision annotated with the value a lie about it would earn,

and assigned the cheapest regime whose cost-to-corrupt clears that value. The oracle for a slow parameter on the deepest collateral on Earth and the oracle for a liquidation trigger on a micro-cap should not be the same object. Today they routinely are.

The endgame is an interface, not a monolith. A protocol would declare, per decision, the regime it requires — *this liquidation price needs at least R2; this parameter tolerates R3* — and a competitive market of quant-layer providers, the signers and committees and disputers and provers, would supply verified (f, θ) to that specification. The intelligence gets as sophisticated as finance demands; the trust gets as strong as the stakes require; and the two are decoupled — the only arrangement under which a protocol can have both. The romance of the word "trustless" is what made the boundary invisible. Naming it — and pricing every crossing against what a lie about it would earn — is how on-chain credit finally gets to be quantitative without getting to be fragile.

Notes and sources

This is a mechanism-design argument, not an empirical study; its anchors are the primitives' own documentation and primary post-mortems of named incidents. Loss figures are as reported by regulators or first-party analyses. Two deliberate omissions of number: zk-ML proving times, costs, and feasible model sizes are kept qualitative because the official tooling publishes no benchmarks; and UMA's exact vote thresholds and window are described by mechanism, not quoted, for the same reason.

[1] EVM compute limits and the rationale for oracles — Antonopoulos & Wood, *Mastering Ethereum*, Ch. 11 ("Oracles"); no native floating point, gas-metered computation.

[2] Pyth pull oracle — first-party publishers, signed updates verified on submission, 400 ms feeds — `https://docs.pyth.network`.

[3] Chainlink Off-Chain Reporting — committee agrees a report, quorum threshold-signs, one transaction carries the median — `https://docs.chain.link`.

[4] UMA Optimistic Oracle — assert-with-bond, challenge window, dispute escalates to a staked-token vote; trust is economic — `https://docs.uma.xyz`.

[5] zk-ML — EZKL compiles an ONNX model to a Halo2 circuit with an on-chain verifier; proving cost scales with model size, inputs quantized, large models infeasible today — `https://docs.ezkl.xyz` (performance figures deliberately not quoted).

[6] Mango Markets, Oct 2022 — ∼$116M (SEC; ∼$110M taken, ∼$67M returned) — oracle manipulation of the MNGO perp — SEC Press Release 2023-13.

[7] Euler Finance, Mar 2023 — ∼$197M, a missing solvency check in `donateToReserves` plus self-liquidation (a code bug, *not* oracle manipulation); funds later returned in full — Chainalysis post-mortem; Euler recovery blog.

[8] bZx, Feb 2020 — ∼$1M across two flash-loan attacks using DEX spot as the oracle — samczsun, "So you want to use a price oracle".

[9] Aave CRV episode, Nov 2022 — ∼$1.6M residual bad debt from illiquid collateral against a static parameter (a parameter/market failure, not a code bug); attempt failed — The Defiant; Blockworks.

[10] Harvest Finance, Oct 2020, ∼$34M (SlowMist); PancakeBunny, May 2021, ∼$45M (Amber Group); Cheese Bank, Nov 2020, ∼$3.3M (PeckShield) — all AMM/pool spot used as the price feed.

[11] Risk-parameter management — Gauntlet and Chaos Labs simulate and recommend lending parameters, enacted through DAO governance; Chaos Labs' on-chain "Risk Oracles" push updates within governance-set bounds — Aave governance forum; Chaos Labs docs.

[12] Market impact and the square-root law — see *The Dimensional Shift*, `https://rhouzlane.com/dimensional-shift`.