Big No No : Insurance Markets and Unattributable Faults

Abhimanyu Nag • 16 Aug 2025

Special Thanks to Ruizhe Jia from Stanford, Yash from Othentic, Shrisht Fateh Singh from University of Toronto, Rizwan from Columbia and Abhishek and Dhruv from Catalysis Labs for their helpful comments and suggestions

> TL;DR: Carelessly designed consensus insurance markets can be disastrous where attackers use the policy itself to lower their cost of corruption, making the system incentive-incompatible and kamikaze attacks like the one considered here are usually not independent. A system-wide event could potentially wipe out everyone at once and insurance then turns into redistribution where premiums explode and security collapses.

I'm not a writer, but I often come across ideas and protocols that promise users safety while unknowingly setting the stage for financial avalanches of attacks and losses. One such idea is the notion of consensus insurance markets and this note is a motivation for crypto people to stop and think about what they're doing before they try to sell this idea to just about anyone for funding money.

Consensus? Insurance? What?

As a primer for the degens, consensus is the process by which a set of nodes (validators, replicas, or agents) agree on a single value or sequence of events, even in the presence of faults. In blockchains, this means agreeing on the next block and in classical distributed systems, it means agreeing on the state of a replicated database. The two types of faults consensus algorithms handle are 1.Crash faults where a node simply stops responding and 2. Byzantine faults where a node behaves arbitrarily, possibly maliciously.

Protocols like PBFT (Practical Byzantine Fault Tolerance), Tendermint, and Nakamoto Consensus are designed to tolerate these, usually up to a fraction of nodes failing (e.g. $f < \frac{n}{3}$). Problem is these systems assume fault attribution: if a node misbehaves, we can identify it and punish it (e.g. slashing in proof-of-stake).

Unattributable Faults : How Would They Even Happen?

Unattributable Faults as failures in consensus where no specific party can be blamed. Let us motivate this with an example. Imagine a network of n validators in a PoS chain. An attacker precomputes a set of conflicting blocks and times a broadcast to hit all honest validators simultaneously, exploiting network latency. Each honest validator, observing the blocks in slightly different orders, finalizes a chain that conflicts with others. When slashing rules are applied, every validator including the attacker who triggered the ambiguity, incurs a penalty proportional to their stake S_i, say $\alpha \cdot S_i$. Formally, the loss vector is

$$L_i = \alpha S_i \quad \forall i \in \{1, \ldots, n\}, \quad L = \sum_{i=1}^{n} L_i$$

and because the attack exploited timing and network uncertainty, there is no provable way to attribute the losses solely to the attacker; all participants are penalized. This is basically a kamikaze attack : the attacker causes maximal system-wide damage while hiding behind honest-looking behavior.

Can this happen today?

This exact scenario is nearly impossible in PoW Bitcoin due to probabilistic finality. Forks resolve naturally via longest-chain rules, and slashing doesn't exist however for PoS Finality Chains like Ethereum it is technically feasible in small or poorly connected networks. Validators must finalize blocks locally and slashing applies for conflicting finality and large, well-connected validator sets make precise kamikaze attacks difficult also network partitions or propagation delays are limited but could be exploited in extreme cases.

Interestingly, in 2010, Bitcoin experienced a temporary consensus fork when two miners found blocks almost simultaneously, creating competing chains. The network resolved the fork automatically, but for a short period, there was ambiguity over which chain was "correct," and miners on the losing chain suffered temporary losses. Similarly, in 2020, Ethereum 2.0 testnets faced network partitions where validators finalized conflicting blocks due to propagation delays. Validators were penalized according to protocol rules, but no single actor caused the divergence.

Enter Insurance Markets

One natural reaction is to try to “insure” against these faults. After all, if validators can lose stake due to unattributable failures, why not pool risk through an insurance market? In theory, each participant could pay a premium to cover potential slashing events, and the pool would compensate those hit by consensus ambiguities. Simple integration right?

But like all these insurance agencies will tell you, the core of any insurance contract is attribution: you must be able to identify who suffered a covered loss in order to pay them. In the context of consensus insurance, this means knowing exactly which validator or set of validators triggered a slashable event. Unattributable faults break this fundamental requirement. When every validator loses stake simultaneously due to a kamikaze attack, network partition, or protocol divergence, there is no cryptographic evidence that singles out a claimant. Without a provable mapping from loss event to responsible party, the insurance pool cannot determine who is owed compensation. Any attempt to pay “everyone” becomes a pure redistribution of losses rather than insurance, and in a system where losses are perfectly correlated, it risks depleting the pool entirely.

Some Cost Economics

Napkin Economics for Attacker

Assume I am the kamikaze attacker. I have X amount staked on a PoS finality chain which follows say PBFT ($\frac{2}{3}$ honest majority) with total stake say T. I go to an insurance company (let’s call them Stupid Insurance) and get my X covered in return for a perpetual premium rate p for time t. I conduct a Kamikaze attack as defined above at time t_{attack} and I lose all my stake in the process and so does everyone. Let my utility be a function of everyone’s losses (which is basically T) and the amount it cost me to conduct the attack. Assume people losing money makes me very happy*.

[enter joker meme]

My utility before insurance is

$$U[\text{attack}] = T - X - p \cdot t_{attack}$$

But with the insurance,

$$U[\text{attack}] = (T - X - p \cdot t_{attack}) + X = T - p \cdot t_{attack}$$

Therefore my cost of corrupting the protocol is just the premium I pay to Stupid Insurance until I decide to conduct the kamikaze attack (which could be the very next day after I get my insurance). Thus technically speaking if $t_{attack} \to 0$ then

$$\boxed{U[\text{attack}] = T}$$

which is pure profit and my cost of corruption is ~ 0. This violates the fundamental condition of cryptoeconomic security where the profit from corrupting this protocol >> the cost of corrupting this protocol.

OPEN PROBLEM: Is there a way to solve this problem and disincentivize me from conducting the attack? Given I am not allowed to stake until a certain period of time (i.e $t_{attack} >> 0$), we need the following inequality

$$U[\text{attack}] < 0 \implies T < p \cdot t_{attack}$$

But we also know that from traditional insurance pricing literature,

$$p \geq E[L_X] + \text{margin}$$

Whoever wants to deal with this be my guest.

Napkin Economics for Stupid Insurance

If Stupid Insurance decides to insure the entire stake value of the chain (i.e.T), it faces the possibility of perfectly correlated losses due to unattributable faults like the kamikaze attack. Traditional premium pricing requires that the premium rate p satisfies:

$$p \geq E[L] + \text{margin}$$

,

where $E[L]$ is the expected loss from the insured event. But in this scenario, $E[L] = T$ for a kamikaze attack, because all validators including insured ones lose their full stake. This implies the premium must be at least equal to the total insured value:

$$p \geq T$$

Effectively, the insurance company would need to charge a perpetual premium equal to the entire stake of the chain to remain solvent in expectation. Any lower premium exposes the company to unbounded liability, since a single unattributable fault could wipe out the pool.

In practice, this makes the insurance economically infeasible: either the premium is so high that no rational validator would buy it, or the insurance pool is underfunded and increases systemic risk instead.

Formally, let C be the capital of the insurance pool. If the unattributable fault occurs, the pool must pay

$$L_{Total} = \sum_{i=1}^{n} L_i = T$$

If $C < T$, the pool cannot fully honor claims, effectively creating a secondary financial contagion where everyone will lose all their stake and there will be no way to recover it.

Thus, insuring against unattributable faults violates the fundamental law of cryptoeconomic security: the cost to corrupt the protocol is supposed to exceed the profit from corruption. Here, the attacker can exploit the insurance mechanism to reduce their effective cost of attack to nearly zero, while inflicting maximum damage on the network.

OPEN PROBLEM: Is there a way to design a safer insurance mechanism that will solve all of this? The problems considered here are tail events which are, while improbable, equally important to normal market conditions.

Conclusion and Future Steps

Now let ChatGPT give you a nice sendoff :: Unattributable faults create a fundamental vulnerability in consensus protocols: losses can occur system-wide without any party being accountable. Insurance markets fail to mitigate this risk because attribution is required to determine payouts, and perfectly correlated losses undermine diversification. Attacks like the kamikaze scenario show how an attacker can exploit these gaps, making the cost of corruption negligible while inflicting maximum damage. Addressing this challenge requires protocol designs with stronger fault attribution, careful economic incentives, and adaptive risk management to ensure that both technical and financial security are preserved.

But seriously though, I am working with folks at Catalysis Labs to solve this problem and we'll come out with a paper soon.

I have been Abhimanyu Nag

*I may figure out a way to capture some of that money through exogenous tools