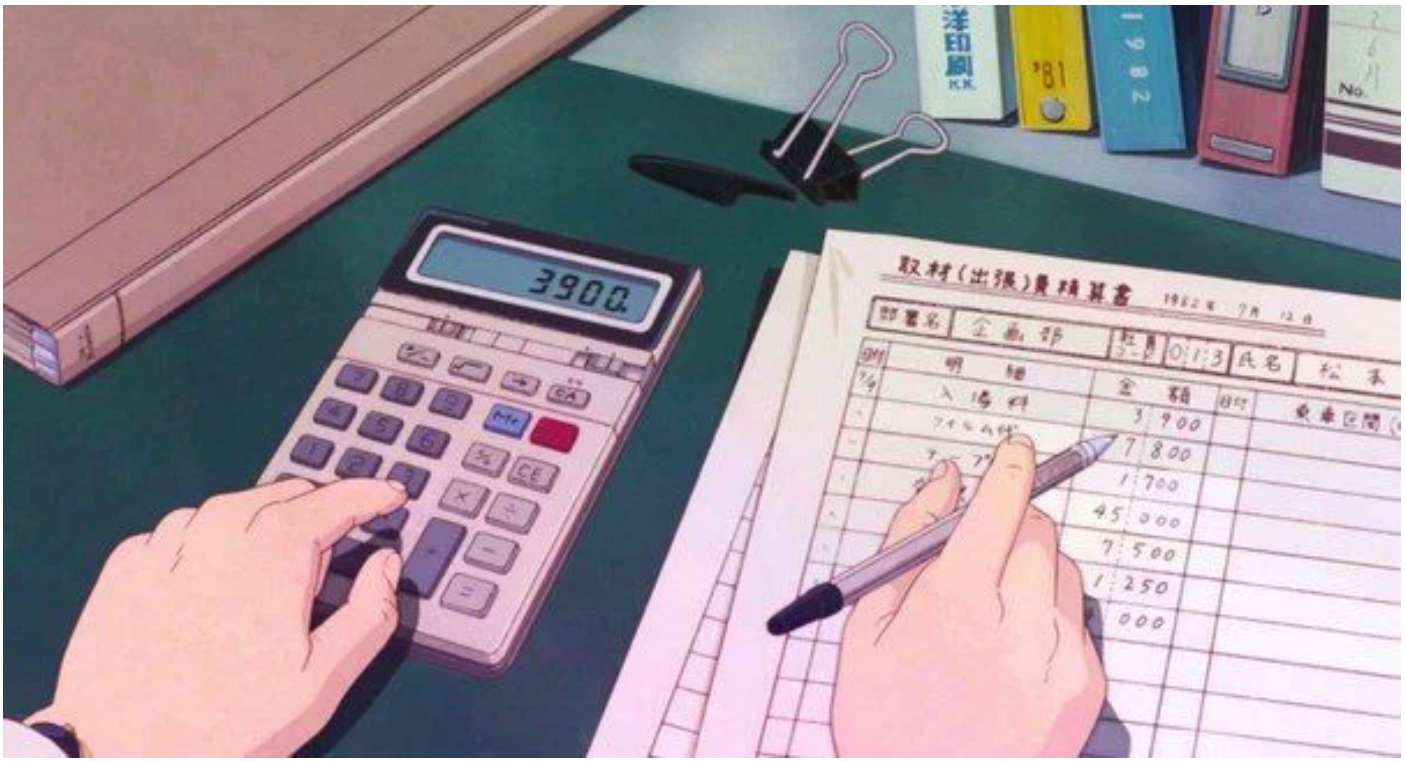
Algorithmic Insurance

0xe16c August 20th, 2021



The idea of Decentralized Insurance has been one of the classic examples for ideal use-cases of blockchain applications. Insurance providers may have incentives to withhold claims for as long as possible, and users have little transparency into the process — leading to an overall poor user experience. The idea of putting these insurance processes on-chain is an appealing one: insurance claims can be programatically & immediately paid out if some conditions are met, and everyone has full transparency on the state of the entire system at all times. For example, someone could buy travel insurance for their flight, and a smart contract could automatically payout if Google Flights reports that the flight was delayed.

Many attempts at decentralized insurance have been tried over the years, but anything that touches the real world has not gotten much traction at all. Firstly, there are many concerns around getting *trustworthy* data on-chain — if a contract can be maliciously fed data that says a hurricane happened when it didn't, that on-chain weather insurance market is as good as useless. Secondly, the current set of Ethereum/DeFi users are not generally looking for vehicle/household/travel insurance on-chain — they are looking for

something more *crypto-native*.

Smart Contract Insurance

The immense growth of DeFi in the last two years has single-handedly revived market demand for decentralized insurance products. Many individuals or funds have significant amounts of capital in these DeFi protocols, and *actually* want protection against smart contract exploits. Hence, a new wave of decentralized insurance protocols have sprung up in the last year, mostly focused on smart contract insurance.

Most of these projects tend to innovate along two main axes — Payout Mechanism and Pricing:

Projects like Nexus Mutual rely on a set of humans to determine if a smart contract exploit actually happened or not, and hence if payouts should happen. Others like Cozy Finance, Ante Finance, and Risk Harbor rely purely on specific on-chain conditions to be met to determine if a payout should happen.

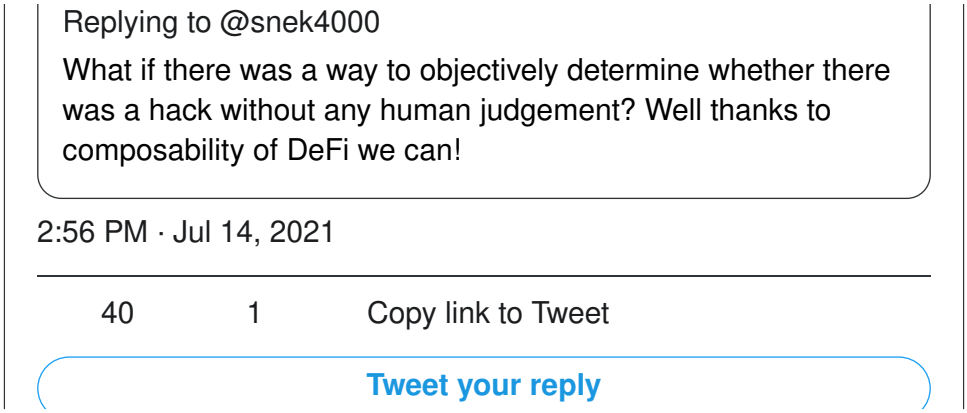
On the pricing axis, some projects use Bonding Curves based on utilization rates to price insurance — e.g some formula that prices the cover based on the demand & supply for that insurance market. Others rely on a team of human experts to determine the "fair" value of coverage for a protocol based on information such as track record of the protocol not getting hacked, audits, and so on.

This post will focus on the Payout Mechanism of decentralized insurance protocols.

Algorithmic Payouts

In traditional finance, there would need to be all sorts of trusted intermediaries to verify if some event happened or not before an insurance company can pay out a claim. However, because smart contract exploits are purely on-chain events, we should be able to write programs on-chain that determine if the other on-chain events happened or not — zero trust assumptions required.

> **Sisyphus**
> @0xSisyphus
>
> Algorithmic insurance is probably the first interesting thing in the defi space I've seen in a while which isn't 1) a new Ponzi game and 2) is harder to build in legacy finance than on a



As builders and investors in this space, we should always be on the lookout for things that blockchains uniquely enable which cannot be done in traditional finance — algorithmic insurance could be one of them.

How do they work?

The easiest way to think about how these systems work is that they are a prediction market on whether or not a certain condition within a smart contract is met. For example, if you lend USDC on Compound, your USDC balance should monotonically increase over time as interest is accrued — if you are suddenly unable to claim less USDC than you originally deposited, one could infer that a smart contract exploit has occurred. Since we can check this on-chain reliably, we can then create a prediction market as to whether or not `getPricePerFullShare() > 1` returns true or false.

Because Compound has never been exploited, we can imagine that most people would prefer to take the "true" side of the market — skewing the odds of the prediction market significantly. This creates attractive odds for the "false" side of the market — if by some chance Compound does get exploited they will get a huge payout, otherwise they will lose a small amount of capital. You can see how this starts to look like an insurance market, where the "false" bettors are paying a premium to the "true" bettors for a chance of a big payout if the prediction market resolves in their favor.

For more complicated DeFi protocols, we may want to define more complex payout conditions than simply checking if a deposit token can claim sufficient underlying asset. For example, we could combine multiple conditions together, such as `totalAssets() > x && getPricePerFullShare() > 1` or even define a condition that checks the

contract state over some time period, such as `totalAssets` must not decrease by more than 50% within a 10-minute window or the condition will return true. The beauty of programmable finance is that we can create arbitrarily complex payout conditions because we know for sure that the on-chain checks will execute deterministically. Furthermore, since all these payout conditions are defined on-chain, we can *always* know when a market will pay-out, no matter how complex the conditions get — it is impossible to obfuscate the risk through legal "terms and conditions".
One way to bootstrap these markets is for the core team themselves to create these insurance markets and take the short side — both as a way to bootstrap liquidity for buyers of cover as well as indicating confidence in their own code. Ante is an example of a project that focuses on this and tries to get adoption through developers writing insurance markets for their own projects. We could imagine a world where projects who do not write insurance for their own protocol are as risky as projects that have unaudited smart contracts.

Potential Pitfalls

When users buy smart contract insurance, they simply want to get covered if a protocol gets exploited. However, it can be potentially difficult to define through code *what* an exploit actually is. For example, a classic "rug pull" where the creator of a liquidity pool removes all the liquidity could lead to a user holding a bag of tokens that are worth $0. Is this an exploit, or is this expected behavior for providing liquidity in an AMM?

One "solution" to this problem is by simply letting the free market determine which markets are the canonical insurance markets for various protocols. For example, we may be able to create two different insurance markets for Compound: the first pays out if cTokens cannot claim >1 underlying, the second pays out if cTokens cannot claim >0.5 underlying. In my opinion, it is likely that the market will naturally converge around one of the two as the "canonical" insurance market for Compound, maybe through Schelling points like the Compound team advocating for one or the other — letting most of the liquidity stay within one market. It could be much less "obvious" which market should be the canonical market for a protocol if the complexity of it is high, but this free market approach assumes that the market will sort itself out.

In this model, it matters *a lot* which specific insurance market you are a buyer or seller of. It is possible that a protocol was obviously (to the human eye) maliciously exploited, but the payout condition that your market resolves on does not get triggered. Conversely, it is possible that insurance sellers may be "unfairly" forced to pay out a claim if an edge case in the protocol causes the payout condition to trigger even in the absence of a malicious exploit.

Hence, accurate communication about what the payout conditions entail are extremely

important so that insurance buyers and sellers know exactly what they are signing up for. This is mostly a UX challenge, and translating the payout conditions from code into words is a non-trivial task. Teams may be tempted to obfuscate the risks of insurance selling as "risk-free yield" to attract TVL, or present the products to buyers as all-encompassing smart contract insurance to boost their volumes.

Beyond Insurance

Although the market for smart contract insurance is huge, this type of protocol can also be used to create markets for exotic swaps aimed towards speculators. Changing the payout condition from things that "should never happen" to "should *rarely* happen" makes it much more interesting for speculators who are looking for asymmetric opportunities.

Composability also allows developers to combine these markets in arbitrary ways, for example rehypothecating the collateral used to sell insurance in one market as collateral in a different market, letting one pool of capital sell insurance on multiple (hopefully uncorrelated) things at the same time.

These tokens from both sides of the insurance market should also be composable with other DeFi protocols. Some ideas include:

- *Protected-cToken,* which is a token that packages the insurance token + cToken into one
- *Perpetual insurance buying,* where a contract can constantly extract yield from a yield-bearing asset to pay for insurance
- *Levered insurance selling,* where users can borrow against their insurance-selling positions to sell more insurance and generate more yield

Algorithmic Insurance opens up the design space significantly for interesting financial products on-chain because it is purely self-contained and does not rely on human decision making to payout. This objectivity helps developers reason about exactly what exposure the insurance market provides, and can build other apps on top of these primitives without worrying about human decision making or trust — which is what DeFi should be all about.

Thanks to Miyuki and Emily for reading this beforehand and giving their feedback

ARWEAVE TX ↗	EwrzHZ…BUJhgU
ETHEREUM ADDRESS ↗	0xe16c…B99CCd
CONTENT DIGEST	72Oo_g…wmMEDU