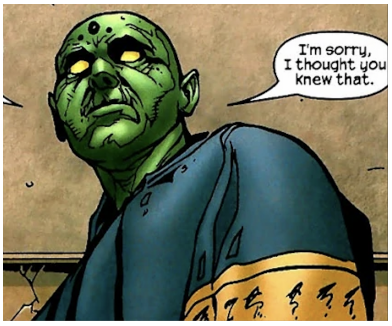
SIWA: Overlay V3

...

March 7, 2022

Abstract



1 Oracle Problem

2 SIWA

Let us simplify matters and consider a pure derivatives market, assuming that whenever an investor executes a long or short position they do so out of *belief in mispricing* — i.e. that the price will increase or decrease respectively. The size of the position is proportional to the magnitude of both the belief and mispricing.

Then, each trade becomes then an implicit *statement* about the future price. Let the time difference between now and the future be Δt. Uncertainty increases with Δt. If Δt is zero, uncertainty is essentially zero. As both risk and reward grow with uncertainty, price oracles can be considered as trades with zero uncertainty, and thus zero risk and reward.

A trade with no risk and reward seems like not a trade anymore, which is why thinking about oracles as utilities rather than risk-taking ventures is typically more useful. The conceptual connection between traders and oracles is nevertheless present, and it is this connection that we leverage in SIWA.

We assume that the users of Overlay are all traders. Traders, however, have different risk profiles. One of the primary insights to come out of Overlay V1 is that these different profiles can be fit together to serve each other and stabilize the system. There are currently two such profiles referenced in protocol design docs: *speculators* (i.e. traders or users) and *basis* traders (also called carry traders). In SIWA we add *creditors*, i.e. those with a large amount of capital who loan it out for interest at very low risk. These are like bond purchasers.

Speculators Basis/carry traders Creditors		

The nature of markets is such that making a statement about future price reduces to making a statement about the statements of others about future price. This is the Keynesian beauty contest. When $\Delta t = 0$, the choices of others are in theory trivial to predict by simply making an observation, which is the basis of the Schelling coin idea.

3 Schelling Coin

We use a proof of stake system, in which capital (ETH, OVL, DAI etc) is locked by oracles. Either the required amount is fixed to obtain one vote (e.g 32 ETH) or the weight of each vote is proportional to the locked capital.

We identify four separate classes of oracles.

1. *Trusted*: nodes which are doxxed – KYC/AML applies and they may even have a legal agreement with the Overlay Foundation.
2. *Sponsor*: nodes which lock an extra amount of OVL as insurance to initially sponsor a feed. They receive outsized rewards but their stake is used to pay back the protocol in case the market they are sponsoring is not suitable (i.e. is predictable or manipulable).
3. *Honest*: nodes which stake and serve prices faithfully.
4. *Attacker*: nodes which stake and attempt to manipulate prices.

Let T, S, H, A be the sets of trusted, sposor, honest, and attacking nodes, respectively. For some individual member $T_i \in T$ we write the amount of capital staked by that individual as $f(T_i)$. Then set

$$\hat{T} = \sum_{i=1}^{|T|} f(T_i)$$

for the total amount of capital staked by all trusted nodes.

The main problem with schelling coin is bootstrapping the network to the point that there are enough individual nodes that cheating is almost impossible. We propose a solution to this problem below. The basic idea is to move from pure centralization to pure decentralization in precisely-defined stages, while handling the attacks specific to each stage.

1. Case 1: All sponsors are actually attackers and there are no honest nodes.

   In this case, there is no way of reliably serving prices and such a breakdown points to a deeper problem in the market. If sponsors are attackers, then clearly the market is broken. The same is true simply if sponsors and trusted oracles are disagreeing on the price. In this case, the solution is simple: *the market is automatically shut down.*

2. Case 2: There are trusted and sponsor nodes but attackers and no honest nodes.

   The solution here is to set the cap of the underlying market to either $\alpha(\hat{T}+\hat{S})$ or $\alpha(\hat{T}+\hat{S}+\hat{A})$ where $0<\alpha<1$ (leverage and a specific risk analysis are the main determinant of α). There are two ways that A nodes can move the price.

   (a) If $\hat{A}>\hat{T}+\hat{S}$ then this amounts to a 51% attack and the price can be set to an arbitrary value. To mitigate this risk we should continue to use TWAPs even in SIWA, and probably even cap maximum price moves in a specific period. This is equivalent to a market going limit up or down. If a market goes limit up more than n times in a given period (maybe $n=2$), *the market is automatically shut down.*
   This limits damages to ℓCnb, where ℓ is leverage, C is the cap, and b is the percentage change in price before the market is limit up. (We could just copy the CME here, but on a much smaller per-TWAP period.)
   After the market is shut down, however, the protocol still has the $\hat{A}$ from attackers in the stake. This can be taken from them, so we must simply set parameters such that

   $$\hat{A}>\ell Cnb. \tag{1}$$

   (b) More plausible is a microcheating scenario where attackers gradually shift the price in their favor by delivering *almost*

accurate prices. If p is the fair price and $p^\dagger$ is the manipulated price, then each TWAP period, attackers can siphon a total of

$$\ell C\Big(\frac{p^\dagger}{p} - 1\Big) \tag{2}$$

in OVL from the protocol.
The solution here is the same as before. If we have n periods during which microcheating is possible, *the market is automatically shut down.* So long as we have

$$\hat{A} > \ell C n\Big(\frac{p^\dagger}{p} - 1\Big) \tag{3}$$

then damages can be recovered.

We note that the solution to both of these cases requires two important features to hold: 1) a high degree of agreement in prices between trusted and sponsor oracles, which likely requires prices being taken from centralized locations; and 2) a governing body that can review suspicious events after they occur. (Ideally this is an external paid auditing firm.)

3. Case 3:
4. Case 4:

4 Almost Arbitrary Blockchain Oracles