Increasing Robustness of the Terra Oracle

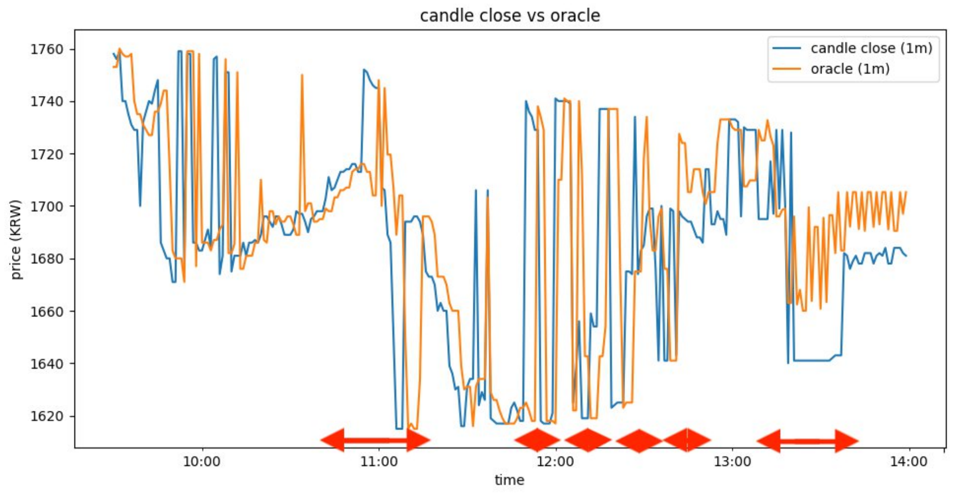
Nicholas Platias
7/26/2019

Following two oracle attacks in the span of one week, we've been debating how to make similar attacks harder and more expensive to pull off. An attacker manipulated the Luna/KRW market on Coinone on 7/15 and 7/22 to create several windows for risk-free profitable on-chain swaps. The vulnerability predominantly lies in the oracle feeders used by validators, i.e. how they interpret market signals to vote on the price of Luna. The oracle is decentralized by design, so it is each validator's responsibility to vote in a way that maximizes the system's security. There is no *one way*, of course. The goal of this paper is to discuss oracle designs that improve on prevailing implementations, and highlight the tradeoffs that arise. Some of those improvements have already been adopted by leading validators. For an analysis of the attacks in Korean read the following two written by EJ — the latest addition to our research team

Two oracles, two attacks

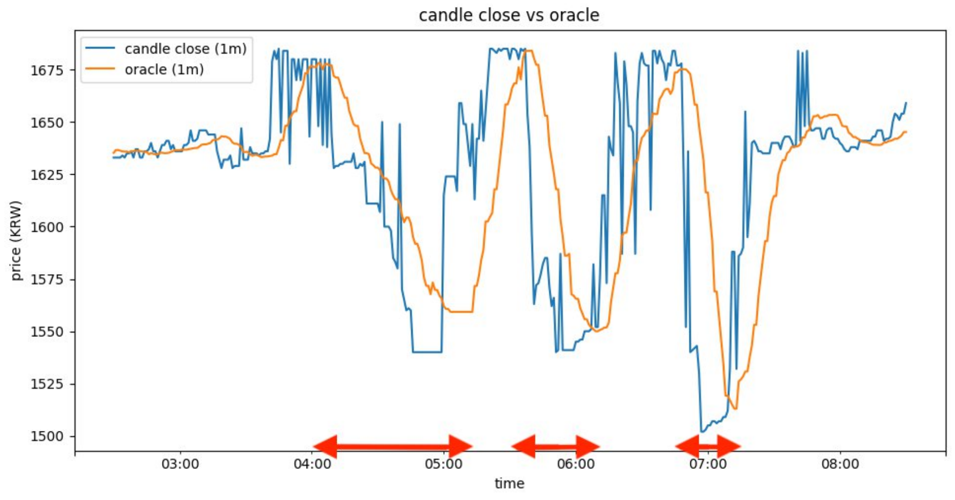
Several factors contributed to the attacks, including the very low liquidity of the Luna/KRW market and the ease of manipulating the market price as a result. The primary enabler, however, was the low cost of manipulating the *oracle* (as opposed to the market). Each attack was able to compromise a different oracle implementation. In both cases, the attacker was able to move the oracle price up and down by several percentage points several times at minimal cost. We review the two attacks and the oracles they successfully compromised.

During the first attack (7/15), the oracle used the closing price of the previous 1-minute candle (latest transaction), so all it took was a handful of transactions at the right moment to move the oracle price all over the place. This is clear in the following graph of the attack, where the oracle price is a short forward phase shift of the 1-minute candle closes.



Evidently, this oracle is extremely cheap to manipulate in a market with a sizable bid-ask spread. In the worst case, the oracle price can be set by a *single transaction*.

Major oracles were updated in response to use a short moving average of closing prices (10-15 minutes). The change made the second attack (7/22) somewhat harder, though not by much. This is clear in the graph below showing 1-minute candle closes and 1-minute oracle prices. The attacker timed the manipulation perfectly, pushing the price either up or down as soon as the oracle had moved sufficiently in the desired direction.



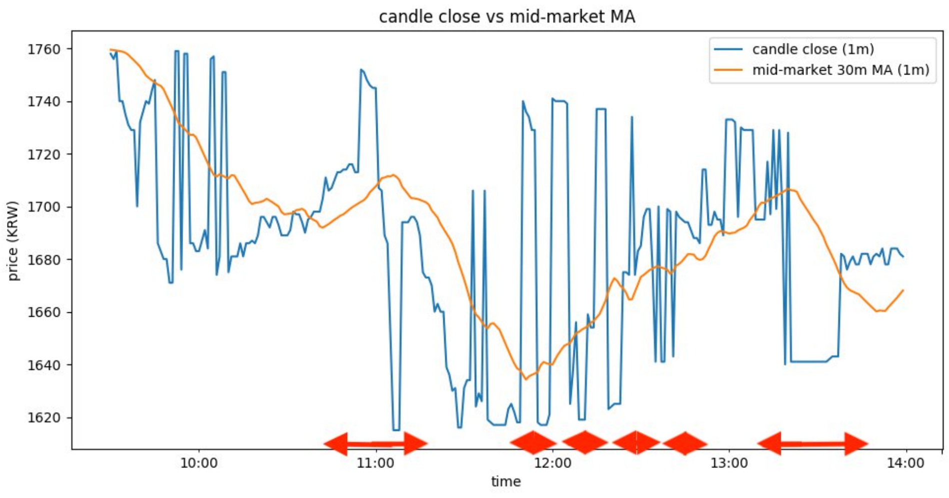
Both attacks proceeded in “rounds” of buy/sell pairs that opened and then closed a position. Complete rounds are denoted in bi-directional red arrows in the graphs above. The first attack consisted of six rounds, the second consisted of only three. The second attack was harder because the oracle responded slower to the manipulated market prices — whereas in the earlier implementation it followed the latest price with minimal lag.

This is evident in the duration of the rounds: average round duration in the first attack was roughly 13 minutes, while in the second attack it was 40 minutes — more than 3x longer. As we explain below, the longer a buy/sell round lasts the higher the risk for the attacker — that is something we naturally want. In this regard, the revised oracle implementation was an improvement — but evidently not good enough.

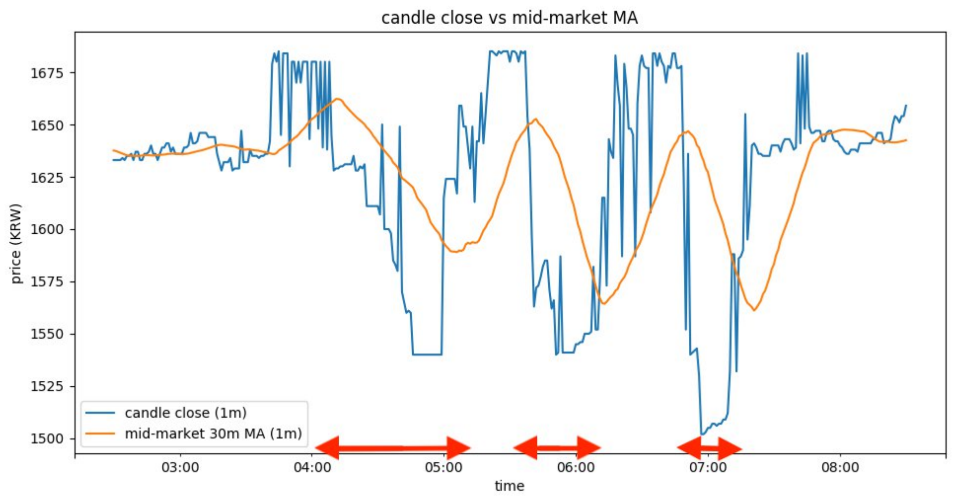
Towards a more robust oracle

Our goal should be to increase the required capital and risk for successful oracle manipulation. There is a critical distinction between manipulation of the underlying market and manipulation the oracle: while we have no control over the former, we need to make the latter significantly harder. Two solutions that have been proposed are the use of *mid-market price*[1] as opposed to closing price, and use of a *moving average* instead of the latest data point. The former is intended to increase manipulation cost: the attacker now needs to control the entire range of trading to set a desired oracle price. The latter, as we already saw, is intended to increase manipulation time and risk: the attacker now needs to control the market over a longer timespan, multiplying transaction costs and increasing the risk that the market will move in the opposite direction.

We simulated a proposed improvement using candle data from Coinone: a mid-market-price oracle that uses a 30 minute moving average. To compare manipulation resistance to the earlier two versions, we plot the proposed oracle during the two attack periods:



[1] Mid-price is typically defined as the average of the highest bid and the lowest ask. In markets where the bid-ask spread fluctuates significantly, a better implementation may be to use the average of high and low price within a given candle.



The mid-market-price moving average oracle is significantly more resistant. In the first attack (first graph), it follows the direction of the market without responding to the erratic short-term closing prices. In the second attack (second graph), it experiences far smoother price moves relative to the existing oracle. It would have been much harder for the attacker to profit in those cases. In fact, if we were to replay the swaps that generated profit (red arrows indicate buy/sell rounds), this time using the mid-market MA oracle, none from the first attack rounds would have succeeded taking into account the bi-directional spread (4%+ in total). Among the rounds in the second attack, only the second and third would have been marginally profitable (< 1% return) assuming perfect timing *if risk is not taken into account*. For comparison, the *actual* return of the third round was north of 5%. It is evident that the **mid-market MA oracle is more expensive and riskier to manipulate: it requires greater control over the range of trading for a longer amount of time.**

A second step towards a more robust oracle is the implementation of circuit breakers. Circuit breakers are designed to curb abrupt price changes and are implemented by most traditional markets. A simple specification for a circuit breaker might be: “if the price changes more than x% in less than y minutes, suspend voting for z minutes”. The circuit breaker can be applied to both market and oracle prices. Appropriately tuned circuit breakers can significantly increase the cost of attacks. It is clear that both attacks could have been stopped with the right circuit breaker. We are actively researching proper circuit breaker tuning.

In summary, recommendations for a more robust oracle are: (1) the use of a 30 minute MA of the mid-market price, and (2) the use of one or more circuit breakers to suspend voting during high turbulence.

Quantifying manipulation cost

Demonstrating that the materialized attacks would have been harder under the mid-market MA oracle does *not* imply that it cannot be manipulated. In fact, a sensible adversary would not have executed the same steps against an oracle under which they would be unlikely to succeed. So the next question to ask is: how much costlier would a successful attack have been? To answer that question, we need a more structured way to reason about manipulation cost. A comprehensive analysis for different oracle designs is outside the scope of this paper — it is something we are actively researching. Here we lay out a simple framework for manipulation cost and briefly discuss how one might approach a more quantitative analysis.

As touched on earlier, we can model the cost of an attack in terms of *capital* and *risk*. Required capital mainly depends on the magnitude of the required market price distortion, the minimum duration the distortion needs to last and the liquidity of the market. The primary risk is that the attacker will not be able to exit his position profitably — this depends on the minimum duration of the distortion, as well as the volatility of the market.

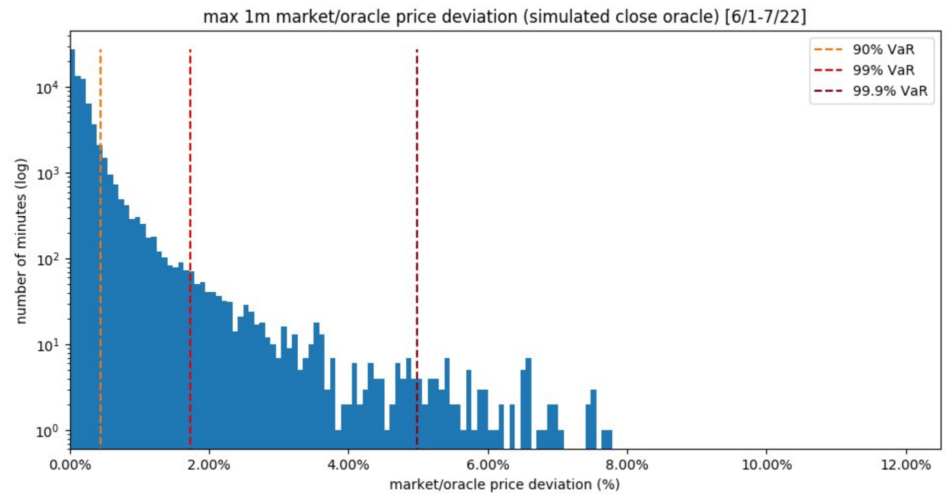
We examine the above factors more closely, and discuss ways to quantify them. The magnitude of required market price distortion depends first and foremost on the *oracle* price manipulation threshold for an attack to be profitable — 2x the on-chain spread (4-20% in total depending on trade size).The oracle mechanism determines how much and for how long the *market* price needs to be distorted until that threshold is crossed. For example, an oracle that uses mid-market price might require a 2% market price distortion, in the worst case, to reflect a 1% change. Similarly, an oracle that uses a moving average imposes an increase either in the magnitude or the duration of a market price change. For instance, for an oracle that relies on a 20 minute moving average to reflect a 1% change, the market price would need to change by 1% for a full 20 minutes, or by 2% for 10 minutes and so on. The principle is that **the harder it is to manipulate the oracle relative to the market price, the more expensive and riskier the attack.**

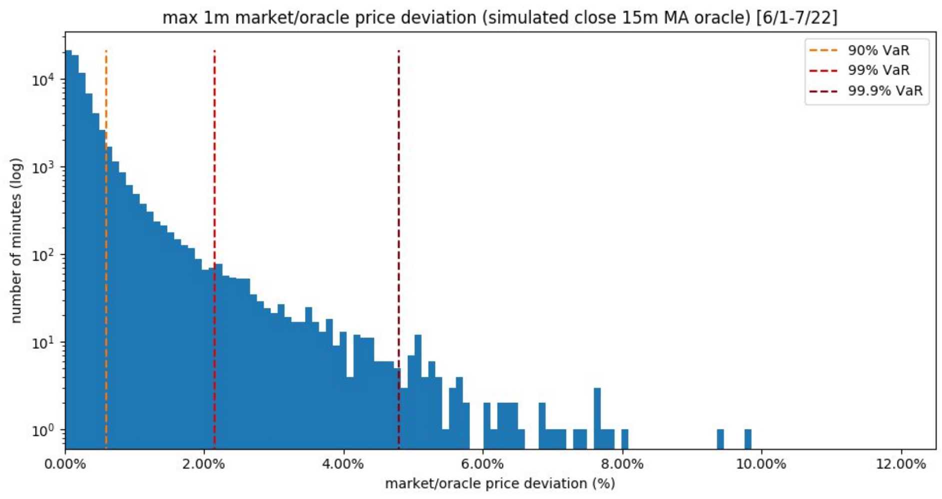
Translating those factors to capital cost and risk comes down to quantifying liquidity and volatility of the underlying market. The two key questions are: how much will it cost me to move the market by x%, and what is the probability that the market will turn against me during the attack window? The lower liquidity and volatility, the cheaper and less risky it is to execute profitable manipulation. An empirical analysis of Luna’s trading data can provide answers to both of those questions. One can then compare oracle designs based on expected capital cost and risk. Unfortunately, Luna’s low liquidity has demonstrated that it is possible for an attacker to exert significant control over Luna’s price using limited capital. That said, we expect that the mid-market MA oracle requires a fair capital multiple and higher risk tolerance for a profitable attack compared to the previous implementations.

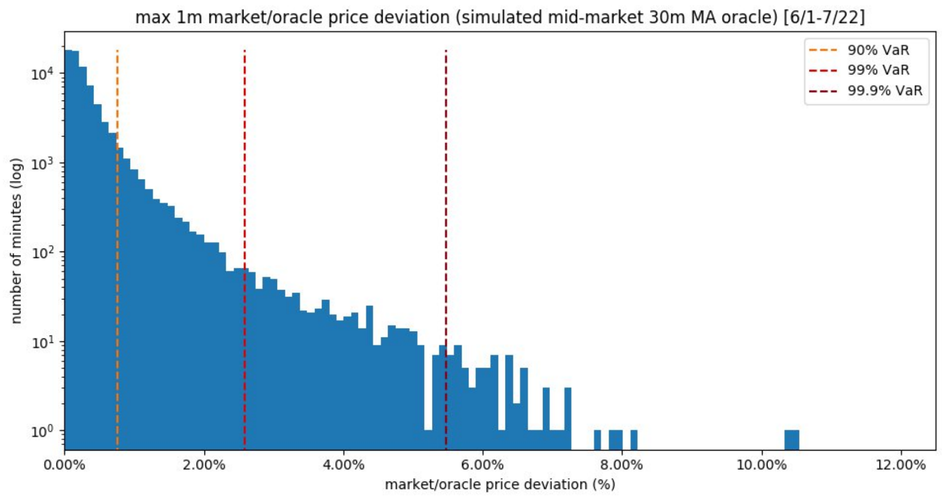
Price responsiveness and arbitrage

A related concern in oracle design is price responsiveness — how fast does the oracle reflect the latest market price? Responsiveness is important for preventing Luna-based arbitrage between on-chain and off-chain markets. Risk-free profit is available whenever the price of Luna offered on-chain diverges from that offered off-chain by more than the combined spreads. Importantly, this is profit that does not directly benefit the stability of Terra, and so is undesirable.

To quantify price responsiveness of an oracle, we measure the maximum divergence (in absolute value) of Luna's price in any given minute from the oracle price. The lower the divergence, the more responsive the oracle. We evaluate the three oracle designs discussed earlier: a naive closing price oracle (with 1 minute lag for the pre-vote), a 15 minute closing price MA oracle and a 30 minute mid-market price MA oracle. We simulated the three oracles on Luna's historical trading data to analyze their responsiveness. Simulation results for value-at-risk of 1 minute market/oracle price divergence are shown below for three different risk thresholds. This analysis is based on Luna's trading history on Coinone since 6/1 (roughly 70K 1min candles of the Luna/KRW pair). Note that the histograms are logarithmic and deviations are reported in absolute value.







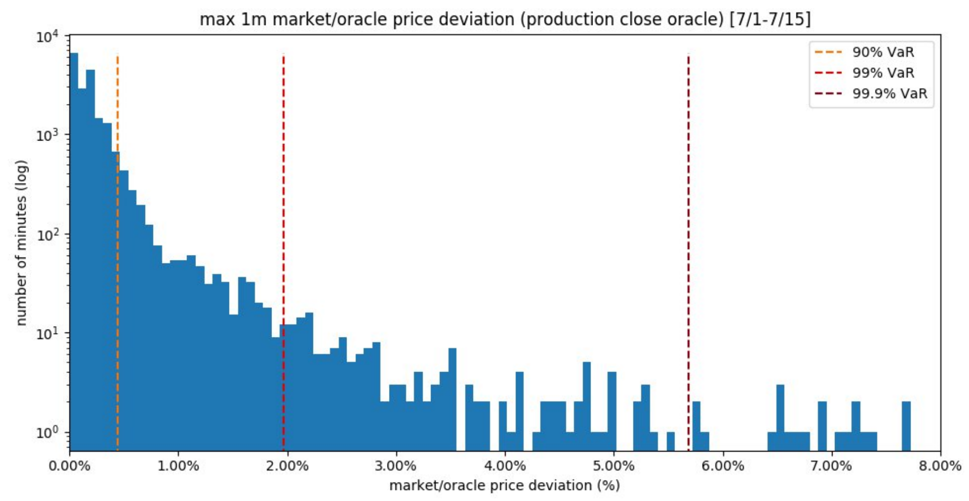
The way to read the first histogram, for instance, is that 99.9% of 1-min candles in the simulation displayed deviation between market and the close-price oracle of no more than 5% (dark red line).

Value-at-risk deviations for the three simulated oracles are summarized in the table below:

		oracle (simulated)		
		close price	close price 15min MA	mid-market 30min MA
market/oracle 1min deviation VaR	90%	0.4%	0.6%	0.8%
	99%	1.7%	2.2%	2.6%
	99.9%	5.0%	4.8%	5.5%

The first thing to note is that deviations tend to increase as the calculation period for the oracle price increases (e.g., deviations for the mid-market 30min MA oracle are larger than for the other two). The differences are modest, nonetheless, especially in the highest risk threshold where deviations for the three oracles fall within a 0.7% range. An exception to the earlier observation is that the 15min MA oracle has a marginally lower 99.9% deviation than the close price oracle. This is likely explained by the one minute lag in the close price oracle, which means that it is sometimes caught far from the action relative to a short MA when the market price exhibits rapid short-term oscillations. Interestingly, maximum responsiveness does not guarantee minimum deviation. Second, it is key to note that the reported deviations are simulated and do *not* take into account the arbitrage opportunity that they often create. If Luna's price were to trade 5% higher than the oracle price, one could easily earn handsome risk-free profit by buying Luna from the protocol and selling it on an exchange. Arbitrage exists whenever the deviation exceeds the combined transaction costs (spread on and off-chain). Therefore, the proper way to interpret those deviations is as "pre-arbitrage" — how much would the market price deviate from the oracle price *if there was no reconciling force in the form of arbitrage*? The natural next question: what do *actual* deviations looks like?

To answer this question, we analyze oracle prices during two distinct periods: 7/1-7/15, the period leading up to the first attack during which the production oracle was implemented using last close price (by the majority of validators), and 7/15-7/22, the period between the two attacks during which the oracle was implemented using a roughly 15min close price MA (again, by the majority).

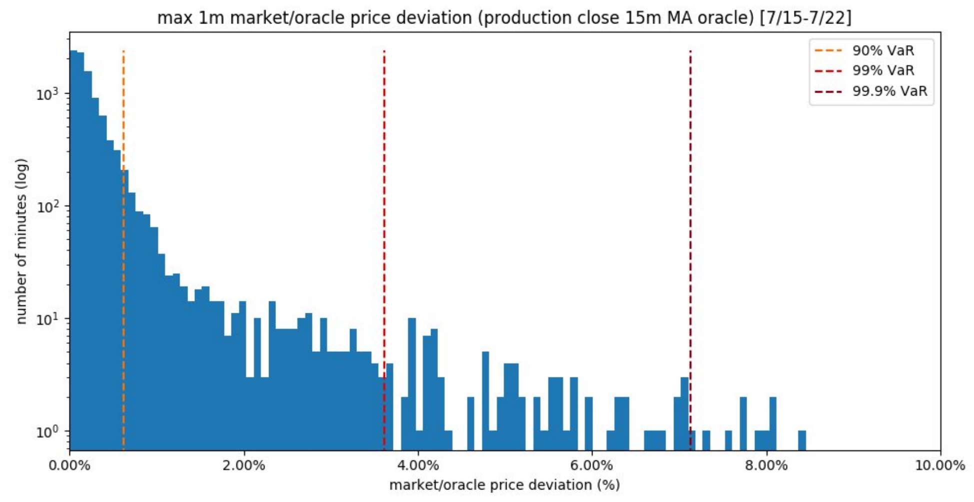


The histogram above shows deviations for the production oracle (price-close) between 7/1 and 7/15. Perhaps surprisingly, deviations are significant. For instance, roughly 1% of 1 minute candles had a market/oracle deviation of more than 2%, which is the minimum spread the protocol charges for on-chain swaps. The majority of those deviations must have led to an arbitrage opportunity, which was evidently not exploited. To properly evaluate the reported deviations we need to compare to the simulated version of the same oracle during the same period. The table below summarizes the comparison between the simulated and production price-close oracles for the period 7/1-7/15:

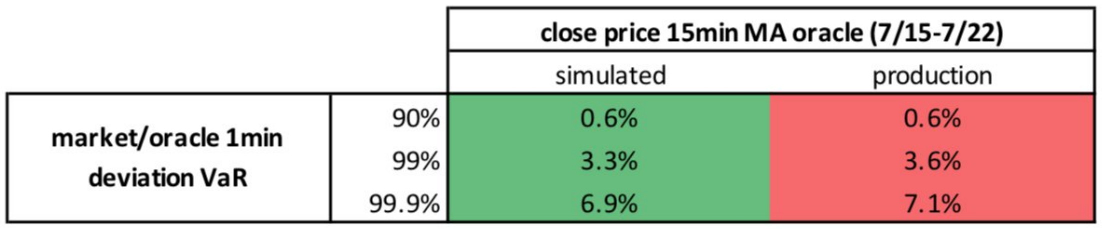
		close price oracle (7/1-7/15)	
		simulated	production
market/oracle 1min deviation VaR	90%	0.4%	0.4%
	99%	1.8%	2.0%
	99.9%	5.3%	5.7%

The production oracle in fact displays *larger* deviations than its simulated counterpart! In other words, "post-arbitrage" deviations are larger than "pre-arbitrage" deviations. This is certainly not what we should have expected: it demonstrates that virtually *no arbitrage* has been keeping the market and oracle prices in check. The marginally higher production oracle deviations might then be explained by variance in price sampling and implementation among voting validators.

We repeat the analysis for the 7/15-7/22 period:



The histogram above shows deviations for the production oracle (price-close 15m MA) between 7/15 and 7/22. Again, deviations are significant. In this case, 1% of 1 minute candles had a market/oracle deviation of close to 4%, twice the minimum on-chain swap spread. To properly evaluate the reported deviations we need to compare to the simulated version of the same oracle during the same period. The table below summarizes the comparison between the simulated and production price-close 15m MA oracles for the period 7/15-7/22:



		close price 15min MA oracle (7/15-7/22)	
		simulated	production
market/oracle 1min deviation VaR	90%	0.6%	0.6%
	99%	3.3%	3.6%
	99.9%	6.9%	7.1%

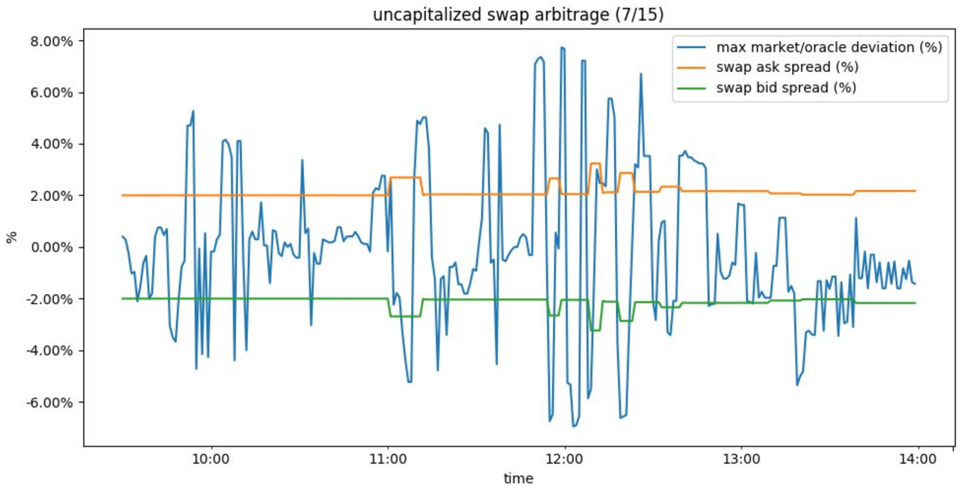
As before, market/oracle deviations are *larger* for the production vs simulated oracle. Evidently, the market has not been exploiting the sizable arbitrage opportunity.

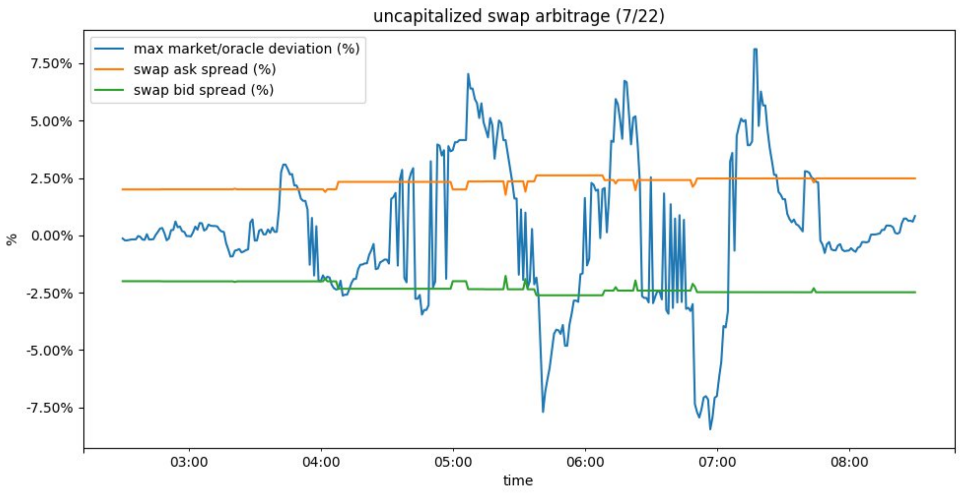
There are three key takeaways from our analysis on oracle responsiveness and market/oracle price deviations. First, deviations between oracle and market price are expected to increase as the oracle calculation period increases. Second, the three oracles we considered mostly follow that pattern but their difference in responsiveness is limited. Third, arbitrage in theory ought to be a reconciling force between oracle and market price, but there is no evidence that it is being exploited in practice.

Oracle arbitrage can increase manipulation cost

Arbitrage between the price of Luna offered on and off-chain is generally problematic, as it impedes price discovery and it incurs a cost on the ecosystem without contributing to Terra's stability. There is a flip-side to this: when arbitrage is exploited, it is a reconciling force between the market and oracle price of Luna, meaning that the market *resists* large deviations between the two, insofar as it is profitable to do so. This is a force that runs directly *counter* to an oracle manipulation: the attacker tries to create large deviations between market and oracle price to direct the oracle in the desired direction as fast as possible. Arbitrageurs are paid to trade *against* the attacker during manipulation, thus increasing capital cost and making the attack more expensive. This is a positive externality of exploited arbitrage.

To make this idea concrete, we review market/oracle price deviations during the two manipulation attacks to demonstrate the opposing interest of attacker and arbitrageur. The graphs below show 1 minute deviations between market and oracle price during the attacks, as well as the live spread for on-chain swaps.





The spread plays a key role here, as it sets the threshold for arbitrage: deviations between market and oracle larger than the spread create arbitrage. Schematically: any deviation (blue line) that crosses above the orange line (ask) or below the green line (bid) offers risk-free profit. For instance, taking cue from the actual spreads, when Luna trades off-chain at a 5% premium, the arbitrageur buys Luna from the protocol and sells on an exchange for 2.5-3% risk-free profit. Evidently, multiple arbitrage opportunities arose during both attacks that were left uncapitalized.

So why are attacker and arbitrageur on opposite sides here? Both graphs above are driven by the attacker, so it is simple to see what is happening: the attacker forces large deviations between market and oracle to direct the oracle price one way or another. In the first attack, this process is repeated at high frequency because the oracle is highly responsive. In the second attack, the process is slower, and the deviation needs to be sustained for significantly longer before the oracle responds. The arbitrageur, on the contrary, profits from closing deviations. If present, the arbitrageur would have traded against the attacker whenever the deviation between market and oracle was pushed above the orange or below the green line. In doing so, the arbitrageur would have cost the attacker a lot of money and would have slowed down the manipulation. Somewhat counterintuitively, had the market eaten its free lunch, it might have also subverted the oracle attacks.

The robustness/responsiveness tradeoff

A key theme that arises from this discussion is the tradeoff between oracle robustness and responsiveness. By robustness we primarily mean *manipulation resistance*. By responsiveness we mean maintaining low deviations compared to the market price, which translates to *arbitrage resistance.* It appears that the more responsive the oracle, the less robust it is against manipulation and vice versa. The tradeoff manifests most clearly when considering the benefits of low responsiveness towards manipulation resistance. First, manipulating an oracle that responds slowly to price changes requires more capital and incurs more risk, as it requires greater control over the market for a longer period of time. Second, the market/oracle arbitrage that is likely to arise from a low-responsiveness oracle invites opposing liquidity and therefore increases manipulation cost.

The key questions are therefore: how do weigh the two against each other, and how do we determine the optimal point in this tradeoff? A simplified way to reason about the tradeoff is as the choice between two evils: being susceptible to manipulation or being susceptible to arbitrage. How do we determine which is worse? I think that the most important variables to consider towards answering this question are *on and off-chain liquidity*. We are currently in a situation where the off-chain market is highly illiquid, and the on-chain market significantly more liquid. Manipulation offers asymmetric returns in this setting: the attacker needs to invest significantly less than he stands to gain if successful. Manipulation is therefore a significant risk. Arbitrage, on the other hand, does not stand to pay much given how illiquid the off-chain market is. This is because arbitrage profit is bounded by the trading volume that will sustain the arbitrage opportunity. All things considered, manipulation risk is a greater evil than arbitrage and so it pays to sacrifice some responsiveness for robustness. The tradeoff would be very different if the off-chain market offered significant liquidity, particularly so if greater than the on-chain market. In that case, manipulation cost would be higher, expected profit would be lower, and at the same time arbitrage profit would be greater. High responsiveness would therefore be top priority. Quantifying the tradeoff is more complicated and outside the scope of this paper. In practice, a good heuristic is to aim for high oracle responsiveness subject to the constraint that manipulation is made expensive and has negative expected return.

Thanks to the Terra validators and community for the active involvement and continuous feedback, particularly the BHarvest team for poking holes and for their great ideas and support.

Thanks to Aiden Lee and Paul Kim for making Columbus data a joy to work with.

Special thanks to terra1q9m7dahs2nrs4v4l83843k9hagxwl0ytqqc8fy for opening our eyes.