OraclΞSwap Technical Appendix

Derivative Contracts for Long-Term Investors

Eric Falkenstein
ericf@efalken.com

V1.0 2/28/2020

This document complements the OracleSwap White Paper by providing an in-depth explanation of concepts and features in the OracleSwap contract. It is meant to be read piecemeal for specific issues, and thus contains some redundancies.

Contents

1 PNL

1.1 Basics

The standard PNL for an investor is straightforward: if you own 10 shares of Tesla stock, and its price moves from \$270 to \$275, you make \$50. More generally:

$$PNL = q \cdot (p_1 - p_0)$$

Here *q* would be the number of BTC, ETH, or SPX contracts, *p* its price is in USD. This is equivalent to

$$PNL = q \cdot p_0 \cdot (p_1/p_0 - 1)$$

This is the payoff we are targeting, the notional amount in USD times the net return. Importantly, the notional and return are both denominated in USD. As we are transacting in ETH, this implies an ETH/USD adjustment at the beginning and end of the return period. For example, in 2018, the Venezuelan stock market rose 80000%, but if you bought a Venezuelan stock, you would have lost 90% of your investment because you need to buy Venezuelan dollars to buy Venezuelan stock there, and the currency devalued by even more. Anyone offering a USD return in a Venezuelan stock will account for this.

The OracleSwap contract is denominated in ETH while its assets are denominated in USD. This implies we must use the ETH price to convert the notional into USD at the beginning of the period, and then convert the USD payoff back into ETH. Thus, contracts denominated in USD will use the ETH price at the beginning and end of the return period to adjust for this.

Specifically, the PNL for these contracts is:

$$PNL_t = RM \cdot LevRatio \cdot \ ETH_{t-1} \cdot \left(\frac{Asset_t}{Asset_{t-1}} - 1\right) \cdot \frac{1}{ETH_t}$$

To be extra clear, it helps to separate this, though we exclude the funding rate:

$$ETH\ notional = RM \cdot LevRatio$$

$$USDnotional_{t-1} = RM \cdot LevRatio \cdot \ ETH_{t-1}$$

$$USDpnl_t = USDnotional_{t-1} \cdot \left(\frac{Asset_t}{Asset_{t-1}} - 1\right)$$

$$ETHpnl_t = USDpnl_t \cdot \frac{1}{ETH_t}$$

Note that the ETH notional, like the required margin (RM), is constant, while the USD notional will change as the price of ETH changes. Thus, if one has 10 ETH notional, the returns will always be around +/- 1 in ETH for a BTC contract, but in USD terms, this could be a lot or a little depending on the future price of ETH. As each subcontract implies symmetric PNLs to Taker and Liquidity Provider (LP), we need only calculate the PNL for one party.

$$PNL^{\text{investor}} = -\ PNL^{\text{LP}}$$

Example 1, BTC: Assume Alice is an LP and posts 100 ETH margin on the BTC contract, a funding rate of 0.25% (7.8% annualized). The BTC leverage ratio (LR) is 2.5. Bob takes the short and posts an RM=10.

> At the Initial Settlement, ETH=$150, BTC=$4000, and the implied notional is thus $3,750.
> Notional = RM·LevRatio·ETH_0=10·2.5·150=3750
> At first Friday settle, ETH=$175, BTC=$5000. Alice is long, so her return is the long return plus the funding rate Bob pays as Taker.
> PNL^{Bob}=Notional·{(P_1/P_0 - 1) - FundRate}/ETH_1
> =3750·{ - (5000/4000 - 1) - 0.0015}/175= - 5389
> PNL^{Alice} = - PNL^{Bob} = 5.389
> The USD return is thus
> Return(USD)= - (5.389·175)/3750=25.15%
> AssetReturn = 5000/4000 = 25.00%

The return difference reflects Bob's funding rate.

Example 2, BTC with ETH changing: Assume everything is the same as in Example 1, except the ETH price is $130 in the second period. At price initialization, ETH=$150, BTC=$4000, and Notional is $3,750 as before.

> Notional = RM·LevRatio · ETH_0 = 10·2.5·150=3750
> At the first Friday Weekly Settlement, ETH=$130, BTC=$5000, and BTC/ETH=0.035
> PNL^{Bob}=3750·{ - (5000/4000 - 1) - 0.0015}/130= - 7.255
> Return(Bob, USD) = (- 7.255·130)/3750= - 25.15%
> AssetReturn = 5000/4000 - 1 = 25.00%

Here in example 2, we see Bob with the exact same position and asset return as in example 1. While the PNL in ETH is different, -5.389 vs. -7.255, the USD PNL and return is the same. ETH movement or level does not affect the USD *return* due to the construction of the contract.

Example 3, SPX: Assume Alice is an LP and posts RM=10 ETH, SPX contract, and the funding rate is 0.04% (2% ann). Bob takes the long and posts RM=10.

> At price initialization, ETH=$150, SPX=$2815. Leverage Ratio of 10, which implies a notional of $15,000.
> Notional = RM·LevRatio·ETH=10·10·150 = 15000
> At first Friday Weekly Settlement, ETH=$155, SPX=$2850.
> PNL^{Bob}=15000·{ - (2850/2815 - 1) - 0.0004}/155= 1.165
> $Return^{Bob}$(USD) = (1.165·155)/15000= 1.20%
> AssetReturn = 2850/2815 - 1 = 1.24%

Again, the return differential is from the financing fee Bob pays to the LP.

If a new position starts on Monday morning, it uses the Monday closing price as its start price as opposed to the prior settlement price. If a position is closed intra-week, say on Monday morning, it uses Monday as the ending price.

1.2 PNL in the Code

The actual PNL calculation in the code is conceptually the same as above. It is applied in a circuitous way because of the memory constraints within Solidity contracts, and also because of the lack of floating-point math (integers are always truncated below).

A subcontract refers to a particular asset within the Oracle contract

Asset: 0 is ETH, 1 is SPX, 2 is BTC

Each asset has a particular leverage ratio, which, due to the lack of floating-point math in solidity, is divided by 1e2 in equations:

Leverage Ratio: 250 for ETH and BTC, 1000 for SPX

At the settlement update, the Oracle contract creates a new set of asset returns from the start of each day of the week to the settlement day. These prices are defined as

Uint[6][3] private prices

prices[i][t] is prices for asset i on day t

prices[1] thus is a 6 element price vector for asset 1: [0, 1, 2, 3, 4, 5]

In a standard week, the 6 slots in the Oracle price vector are:

{0:Friday, 1:Monday, 2:Tuesday, 3:Wednesday, 4:Thursday, 5:Friday}

If there is a holiday on days 0 through 4, the data in slots 0-4 are moved downward.

The price slot-day correspondence if Wednesday was a holiday:

{Friday, Monday, Tuesday, Thursday, Null, Friday}

The price slot-day correspondence if the most recent Friday was a holiday:

{Friday, Tuesday, Wednesday, Thursday, Null, Thursday}

The price slot-day correspondence if the prior Friday was a holiday:

{Thursday, Monday, Tuesday, Wednesday, Thursday, Friday}

For new positions, Price[i][t] is the starting weekly price, and prices[i][5] the ending price. If taken on Friday before the oracle price update, t=5, so the settlement will not assess a PNL for that subcontract.

For rolling positions, Price[i][0] is the starting weekly price, and prices[i][5] the ending price.

For expiring positions, Price[i][0] is the starting weekly price, and prices[i][t] the ending price.

Last week's settlement price, is always [0], while this week's settlement day is always equal to 5, and so if you query the oracle contract and see currentDay=5, you know the settlement prices are the most recent prices in the oracle contract.

On a settlement update, the Oracle contract creates two vectors of 5 returns for each asset. One is for new positions, from the initial day to the end of the week, the other for closing positions, from the start of the week to the closing day.

$$\text{return(start, end)} = \text{AssetPrice[end]} \cdot X / \text{AssetPrice[start]} - X$$

Due to the lack of floating-point arithmetic, the large constant X prevents rounding errors and basically makes this the net return times X. We have X = 1 eth (i.e., 1e18).

Next we adjust for the ETH price, as all subcontracts have an adjustment for the starting and ending ETH price (turning notional into USD, then turning USD PNL into ETH PNL). Here the start price is in the numerator, because we initially multiply it by our ETH notional, and then at the end divide by the ending ETH price to generate the targeted USD returns, in ETH. Note we divide by 100 because leverage ratios are multiplied by 100 (e.g., 250 for 2.5).

$$\text{PNL} = \text{return(start, end)} \cdot (\text{EthPrice[start]} \cdot \text{LeverageRatio} / \text{EthPrice[end]} / 100$$

Lastly, we bound the return at 0.975 of the constant we use to avoid floating-point problems{min, max} = {-X·0.975, X·0.975}. This is because the PNL payments are capped at the required margin. As the financing rate is capped at +/- 0.025·RM, so any returns greater than 0.975·X in absolute magnitude could imply PNL greater than a counterparty's RM. This allows us to multiply a subcontracts RM by the PNL vector and be sure the counterparty has sufficient margin to cover their debit.

As the RM, like the PNL, are in wei, we divide by 1e18 to normalize this back into wei. Then we subtract the subcontract's specific funding rate. First, we turn this into the same units as the PNL by multiplying by 1e18. As the funding rate is in basis points as a percent of RM (e.g., 250 is 2.5% of RM or 1.0% of notional for crypto), we divide by 1e4.

$$\text{SubContractPNL} = \text{RM} \cdot \text{PNL} / 1e18 - \text{subcontract.fundingRate} \cdot 1e18 / 1e4$$

This gives us the PNL for a long taker. The sum of the taker PNLs is then subtracted from the LP's margin at settlement. As each subcontract's PNL is capped at its RM, and the LP's RM is netted, the LP will always have enough margin to pay for any possible debit.

There is a worksheet in the excel spreadsheet OracleSwapData.xlsx with the PNL logic. Note that the web version uses Szabo instead of Ether, so units use 1e12, not 1e18. Users should do a search an replace on 'szabo', changing it to 'ether', and changing 1e12, 1e9, or any 1e10+, and add 6 to the exponentiation.

1.3 Basis Rates

The basis rate is a transfer between short and long, and so does not represent a fee to the oracle or LP, but rather, it reflects the equilibrium demand for long short and long asset exposure.

The *basis* is the difference between a forward and spot price and is defined as the difference between the spot or cash price and the futures price. Alas, while most references define the basis as *spot – future*, and some use *futures – spot*; it does not matter as long as you are consistent. It comes from the basic equation relating the basis to funding rates is based on arbitrage. Say you own USD. You can invest in US Treasuries and get the US interest rate return, $1+r_{us}$ Alternatively, you can buy currency in country A, earn A's interest rate r_A, and then convert this back into USD at a current forward or futures price (these interest rates and futures prices refer to the same duration, so that a futures price in 6 months would use the 6-month interest rates, etc.). This generates the following arbitrage condition:

$$1 + r_{US} = \frac{1}{S_0} \cdot (1 + r_A) \cdot F_1$$

Here S_0 is the current spot price of currency A, in USD per A. F_1 is the forward price of currency A. Rearranging we get

$$\ln(F_1) – \ln(S_0) \approx r_{US} – r_A = \text{basis}$$

This equation is called *covered interest rate parity* and matches forward currency prices and interest rates very well due to arbitrage in these liquid markets. There are specific pages in Bloomberg that show libor and forward currency prices, and they are always an almost exact identity. Without a futures price, a swap referencing a cash (aka spot) price accounts for these costs via explicit funding rates. We can rewrite this formula to apply to a contract that uses only spot prices as follows:

$$\text{Swap Long Return} = (\text{CashPrice}_1/\text{CashPrice}_0 – 1 – \text{basis})$$

Based on interest rates and dividends, the basis for the ETH and BTC contracts should be like that for gold, in the gold does not generate an interest rate while saving in USD does. Gold's basis is the USD interest rate for USD gold futures. Therefore, the long positions in ETH and BTC should pay 1.5% while the shorts would receive 1.5%. For the SPX, we must consider the dividend on stock, which at around 1.8%, is about the US interest rate, resulting in a long funding rate of near 0% for the SPX.

For many assets, especially commodities, interest rates and dividends are of less importance in determining the basis. That is, the complete basis is comprised of the following factors:

$$\text{basis} = \text{interest rate} + \text{storage costs} – \text{dividends} – \text{convenience yield} \pm \text{risk premium}$$

Storage costs include insurance or the cost of decay of agriculture commodities. Convenience yield refers to the benefit of being the owner when there are occasional shortages that cause price spikes, or the benefit of using the good as collateral or a means of payment (and thus, T-bills have an excellent convenience yield). The risk premium is a term for everything else, but reflecting the fear of some catastrophe such as the possibility of a war stopping oil supplies, or that a particular country will decide to devalue their currency. As the convenience yield and risk premium are not as explicitly measured, issues related to convenience yield are sometimes conflated with the risk premium and vice versa.

A commodity like oil has periods where the basis is significantly negative or positive, while the gold basis is entirely explained by the nominal interest rate. One could say that oil has a highly volatile risk premium, but one could also point to more prosaic explanations, such optimism among the futures traders. The bottom line is that outside of interest rates and dividends, the basis is affected by subjective issues that are difficult to quantify but do not constitute arbitrage.

Basis rates are like interest rates, in that they are strongly autocorrelated, much like an interest rate. A futures basis does not whip around nearly as much as the spot price, but rather will tend to be positive or negative for several months. The highly volatile basis we see at BitMEX reflects the unique market there that precludes arbitrage with cash exchanges, and also allows BitMEX to game their basis to maximize revenue. A revaluation has a first-order impact on the price, not the basis.

1.4 Implicit Collar Cost of PNL Caps

The Leverage Ratios imply that the RM will cover and approximate 3-standard deviation event given the notional. For example, an asset with a 100% annualized volatility has a 14% weekly volatility. Thus a 3σ move is around 42%, so a notional position of 100 with margin covering a 3σ event has a margin of 42, implying a Leverage Ratio of 2.4. Note that this implies the expected cashflow—debit or credit—will be around 1/3 of the RM, and investors should anticipate this accordingly (i.e., weekly volatility is about 33% of RM for all assets). The current volatilities for ETH and BTC are around 70%, so the leverage ratios of 2.5 are conservative as a forward-looking estimate of margin coverage.

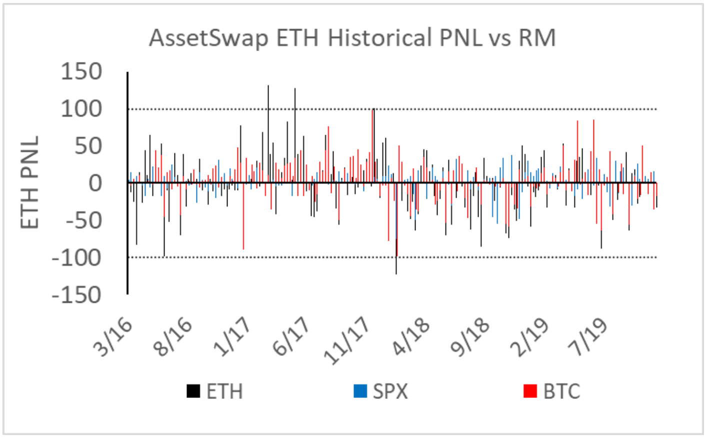
Eliminating all cases where a PNL is truncated requires excessive margin, which is costly, but a margin too small would not meaningfully replicate a true long/short exposure. As with most things, you can get most of the benefits at a fraction of the cost if you tolerate some imperfection.

The RM cap on PNL generates an implicit sort of *collar*, which for the long is a portfolio short an out-of-the-money call and long an out-of-the-money put. The short's collar value is just the opposite as this is a bilateral contract, so the short is then long the call and short the put. If the collar is worth 1% annualized to the long, it is worth -1% to the short.

Unlike most collars, this one has extreme out-of-the-money options. Leverage ratios (LRs) are set so that the margin covers 99.5% of weekly outcomes. Simulating contract PNL using actual data since 2016, only 4 out of 573 weeks generated a PNL capped by the RM, 0.6% of the observations. The outliers were for the ETH contract: twice in early 2017 when ETH doubled in value, and at the beginning of 2018 when ETH fell by one-third. While the SPX margin in this period seems, if anything, overly conservative, this period was one of below-average volatility. Weekly SPX data back to 1950, assuming 100% annualized ETH volatility, a 10% margin (i.e., 10x leverage) generated an insignificant number of violations.

PNL Simulations

Weekly data from March 2016 through October 2019. All contracts have RM=100 ETH. PNL here is in ETH, not USD, as the RM cap applies to ETH.

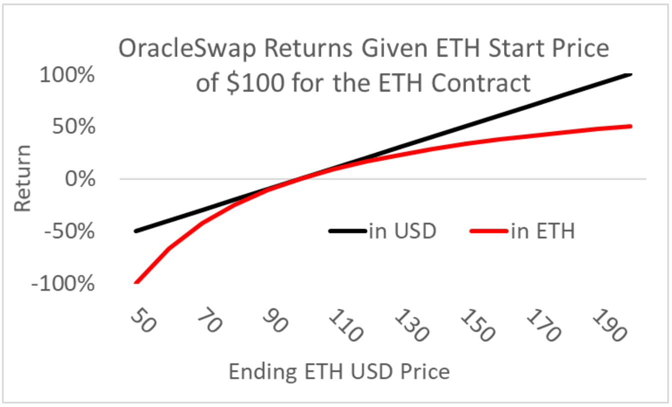


The interesting point here is the unusual nature of an upside truncation in this sample period. Consider the ETH PNL in OracleSwap. For the ETH contract, if we ignore the funding rate, the PNL translates to:

$$PNL(ETH) = Notional(ETH) \cdot \frac{ret(ETH)}{1 + ret(ETH)}$$

For the contract referencing ETH's USD price, an asymmetry comes from the fact that when the ETH loses 33% of its value, the long now also owes 50% more ETH because the USD payout is denominated in ETH. Given a leverage ratio of 2.5, the RM cap is hit when the ETH value falls by 28.6% (2/7) or rises by 66% (2/3). While this may seem strange, the target is the USD return, and indeed the return in USD is linear in the USD Price change, while it is concave for the ETH return as a percent of ETH notional. A decline in ETH price means that to pay the same amount of USD back, you must provide more ETH, so the ETH return is exaggerated on the downside. On the upside, a higher ETH price generating a profit means you need to provide fewer ETH for a target USD payment, implying a lower ETH return. This generates the following concave relation:

Figure 3



Using conservative volatility and correlation assumptions, we can create a forward-looking estimate of collar costs. We generated 100,000 simulations using volatilities assumptions of 70% and 100% annualized for the BTC and ETH, respectively. We assumed a mean return of 0% and that the returns have lognormal distributions. The correlation assumption between the BTC with the ETH was 0.9.

Table 1 shows that contrary to the 2016-9 sample period, the long should anticipate a benefit, but only by at most 0.4% annually.

Table 1

Monte Carlo Estimates of Collar Value

Annualized valuation as a percent of notional

BTC	ETH
0.11%	0.36%

We generated 100,000 simulations using volatilities assumptions of 70% and 100% annualized for the BTC and ETH, respectively. Leverage ratios for these contracts were 2.5. We assumed a mean return of 0%, and that the returns have lognormal distributions. The correlation assumption between the BTC and ETH is assumed to be 0.9.

For the SPX we have data going back to 1950, and there are only 7 weekly returns greater than 10% in absolute value (an LR=10 implies a 10% RM cap). Only two of them were above 11%, both declines by 15% (in October 1987 and October 2008). A Monte Carlo simulation of correlated ETH returns (0.2 with the SPX) does not add any significant information, in that we basically have two 5% truncations to the short's PNL, and this generates a long collar value estimate of +0.25% (~2×5% over 69 years). Note that since we have data on ETH, stock volatility has been below average, so while we have not come close to breaching the RM on our 10x leveraged contract, we want a contract that is robust to the inevitable above-average volatility periods.

In sum, the value to the long from these collars is well under 0.4% annually. In the context of bitcoin funding rates averaging 10% for bitcoin on BitMEX, with monthly averages ranging from +100% to -50%, any explicit adjustment for the effect of the RM cap on PNL is a distraction. The long collar value within this contract is two orders of magnitude less important than the risk premium driving the equilibrium differential in long and short funding rates. It is for this reason PNL caps are not economically relevant as a practical matter.

2 Position Management

2.1 Weekly PNL and Settlement

Settlement transfers the week's PNL between counterparty margins as follows:

$$\text{Investor[j] Margin}_1 = \text{Investor[j] Margin}_0 + \min(RM,\max(-RM,PNL[j]))$$

$$\text{LPMargin}_1 = \text{LPMargin}_0 - \sum_{j=1}^{J} PNL[j]$$

Here the min/max function is the collar on the PNL in that RM is the maximum credited or debited for any subcontract. Note the sum of the investor PNL is debited from the LP so that total ETH in the contract is not affected by the settlement, just its attribution via the various player margins. There is an additional issue related to the burn discussed in section 6.3, but that will be rare (if frequent, no one would use this contract), and is not addressed here.

Players can add to their margin at any time. Payments are immediately credited to the player's margin within the book contract. Players cannot withdraw or cancel during the period between the Oracle settlement update and the LP's book settlement.

A player will know exactly what their PNL will be given the Oracle Price Contract update around 4:15 PM, though they should have a reasonable estimate earlier in the day. Any player who anticipates their current margin, after the PNL is applied, will be below their Required Margin, should cure their position by sending ETH to their margin to avoid default. Cancels will be known with certainty after 4:15 PM, allowing LPs to anticipate their effect on their net RM; defaults, however, are less certain because the player could cure at any time.

During the settlement function execution, each taker is evaluated for having a total margin > RM for their subcontract after the PNL debit/credit. If a player's total margin is less than their required margin upon settlement they default and a default flag is set so that at redemption, 50% of their RM is debited from the taker's account. The subcontract is immediately terminated, removed from the LP's RM calculation, and can never be processed at settlement again.

The settlement function is restricted only to occur at least 24 hours after the Oracle settlement. An LP should be sure to settle between 24 and 48 hours after this time. However, anyone can run it, and once executed in the settlement period, it cannot be rerun until the next Oracle settlement update, so allowing anyone to pay gas to do this is harmless.

If the settlement functions are not completed within 48 hours of the Oracle settlement update, any taker who executes the 'inactive LP' function gets the LP's default fee of ½ their RM (if the LP has less than that in their margin, they will pay whatever they have).

Anyone can settle an LPs book because the settlement can only happen once a week, and if someone wants to pay gas to do this, an LP is strictly better off. LPs should take responsibility for settlement, as a failure to settle on settlement day will prevent settlement for another week. If a settlement is missed because the Oracle was inactive and never posted settlement prices, any player in the LPs book can execute the inactive function, which allows all players to redeem their subcontracts and withdraw their entire margin (i.e., if the book was not settled, and it has been at least 48 hours since the last Oracle settlement update). If the Oracle was active, and no one settled the LP's book on settlement day, the first taker to execute the inactiveLP function will get the LP's default fee as a reward.

Weekly settlement periodicity is the point of moderation between two extremes. If one created a trade that settled in six months, the margin would have to be five times larger to handle the greater expected price variability. The strategy would also then be more like a one-period game, not a repeated one because there would be many fewer future periods in oracle's cheat calculus. If the contract had a daily or intraday settlement, this would require daily attention by investors, and more gas used for margin cures and running the settlement functions.

Real-time margining would also add the cost from having to audit prices referring to odd times, including checking for the absence of price updates when there were large spikes. More importantly, the infrastructure needed to provide real-time pricing is orders of magnitude greater than that required for daily pricing.

2.2 LP Management

LP revenue comes via the funding rate that is part of the PNL, and also the closing fee payment. LP closing fee revenue gets attributed to the LP's margin account.

At settlement, an LPs net margin is recalculated as:

$$\text{LP RM} = \text{Net Margin} = \text{abs}(\text{long RM} - \text{short RM})$$

Within any week, the intraweek LP RM calculation are simply additive: each new position is not netted.

LP's are protected from getting too unbalanced in that takers cannot take positions that push an LP's book beyond 50% of her total offered amount. Intraweek, new positions add to the LP's RM and are not netted. In addition, a book that is well out of 50-50 is constrained so that the LP does not develop a large net position, such as posting 100 ETH and having all of it taken long. Specifically, the rule is:

$$\text{Max Short Take} = \text{Max}\{0,\text{Min}(\text{LP excess marg}, \tfrac{1}{2}\text{ Total LP Margin} + \text{LPShort} - \text{LPLong})\}$$
$$\text{Max Long Take} = \text{Max}\{0,\text{Min}(\text{LP excess marg}, \tfrac{1}{2}\text{ Total LP Margin} + \text{LPLong} - \text{LPShort})\}$$

Table 2 gives examples

Table 2

Maximum Taker Amounts as a Function on an LP's Book

Long Takers	Short Takers	LP Margin	LP's RM	MaxLong Take	MaxShort Take
0	0	100	0	50	50
33	0	100	33	17	67
66	0	100	66	0	34
0	33	100	33	67	17
50	50	100	0	50	50
75	75	100	0	50	50
200	200	100	0	50	50
200	200	100	0	50	50
800	850	100	50	50	0

An LP that fails to run their book's settlement function within 48 hours after the oracle settlement update can be put into default. The first investor to run the 'inactiveLP' function gets the LP's default fee, and all investors can then redeem their subcontracts and withdraw their ETH. If an LP defaults by not having enough margin to cover her required margin, all her subcontracts are canceled at that settlement, and the book cannot accept new investors.

If a player has less than ½·RM in their margin at default (e.g., they have 1.4 ETH the margin of a subcontract with RM=1.0 ETH, and lose 1.0 ETH), the default payment is the maximum of ½·RM and whatever is there. The fact the oracle gets the default fee should motivate the LP to burn instead of default in the case where the LP thinks they were cheated via fraudulent prices. A burn would prevent an evil oracle from receiving both the LP's PNL debit *and* the default fee.

As LPs choose their closing and funding rates, it may be that certain investors prefer low closing fees and high funding rates, while others the reverse. LPs can also differentiate based on their minimum position size.

There are economies of scale in being an LP. The time needed to tend the position is basically fixed regardless of LP size. However, processing 250 investors in one LP uses less gas than if there were 10 investors across 25 LPs since each LP settles their own book independently. Further, the more investors one has, the greater the likelihood the individual book will be flat due to the law of large numbers, and the greater the ratio of total notional exposure to net exposure. Economies of scale, not delusional expectations of future token price appreciation, encourages early LPs, as their cost advantage will build a mote against potential entrants. Competition will lead to lower funding rates and closing fees for investors.

Lastly, there is a managed account contract that enables an investor to delegate LP administrative duties—setting fees, curing margin, adjusting minimum size requirement—while preventing the manager from sending ETH to anywhere but the AssetSwap contract, which can only send its money back to the address that created its position. The investor has sole ability to withdraw funds from the contract. Many brokerage accounts have this capability, where the manager can buy and sell stocks on an exchange, but they cannot withdraw cash. It has a built-in mechanism to apportion a manager fee as a percent of assets under management, and protects the manager's accrued fee as well.

An LP should assess their book for potential defaults and assume the worst-case scenario for settlement. For example, if there are only 3 hours left until the LP can run the settlement function, a prudent LP should assume all potential defaults that lower their RM will not occur, while all defaults that increase their net RM will occur.

Takes are not allowed during the settlement period so that LPs will have a better estimate of their Required Margin.

Book settlement involves three separate functions: the first is for the taker long positions, the second for the shorts, and finally, for the LP's margin adjustment and check of LP margin adequacy. These must be run in succession. See section 8 below.

2.3 Monitoring and Punishing the Oracle

Settlement. Friday is the price settlement day, except for when Friday is a holiday as defined by the New York Stock Exchange, in which case it is Thursday (there are no two-day consecutive NYSE holidays). A counterparty can see if the current oracle prices are the settlement price by checking if 'currentDay==5' in the oracle contract, or if the last Oracle contract update was a settlement update (these are both state variables in the Oracle contract). Players can monitor the settlement by looking at the LP's lastBookSettlement, to see if they are negligent. If a player has a large PNL expected, and thinks the LP may simply not run the settlement function to avoid this payout, anyone can execute the settlement if it has not been run that week.

Burn. A burn should only be invoked if the Oracle prices look fraudulent, and one suspects one's counterparty is part of this conspiracy. For example, if a player—LP or Taker—is long, and the Oracle reports a reference asset price well below the 'true' price, it is reasonable to infer the Oracle is implementing a scam. In such a scenario, the long counterparty would be wise to burn to both inflict righteous justice and avoid exposing one's margin to another week with a cheating oracle.

This involves the payable burn function, where the player must send RM/2, which is effectively burned. The burn prevents that ETH from going to their counterparty, and like the burnt PNL is left in contract limbo, unobtainable by any party to the contract.

For example, if one anticipates having 0.9 ETH in their margin after the PNL, and their RM is 1 ETH, if they pay the burn fee of 0.5 ETH they get their entire 0.9 ETH back. If they do not cure, they would only redeem 0.4 ETH; the other 0.5 ETH would be effectively burned as a default penalty. In this way, a burn and default both cost the same. However, if they have been cheated, a user would prevent the cheater from getting their ETH via the PNL and also the default fee. It generates a burn event log highlighting the oracle's bad behavior.

Takers with less than 50% excess margin, who are cheated by their entire RM amount, will find defaulting cheaper than burning in that at default they are charged 50% of their RM. However, as MakerDAO highlights, most users overfund their margins to avoid having to take emergency measures to cure their margins and prevent default. Though we can only speculate, it is likely most users will also have overfunded margins, and find burning and defaulting equally costly, though with the burn benefit of preventing an objectively bad person from profiting at their expense.

A fraudulent price would almost surely happen at the settlement update because a cheating oracle would gain nothing by posting fraudulent prices earlier, as this would only give its victims more time to react. Thus, settlement prices would be where we expect to see fraudulent prices. After settlement prices are updated, all parties have at least 24 hours to enact the burn function before the LP runs the settlement function on their book.

Inactive LP. If an LP neglects to settle, first, a taker must accept some responsibility in that anyone can settle an LP's book on settlement day from 4 PM Saturday to 4 PM Sunday. Yet it is ultimately the LP's responsibility, and so the LP bears a cost for their negligence. If the settlement period passes without a settlement (48 hours after the Oracle settlement update), the first taker to execute the inactiveLP function receives RM/2 from the LP's margin (if the LP has less than this in their margin, the taker receives whatever is there). The LP is also penalized any amount above that, an amount that is sent to a burn address. At that point, the book is in default, and all players can redeem their subcontracts and withdraw their margins.

Inactive Oracle. After 10 days and there has been no Oracle settlement update after the previous book settlement, this implies the oracle is incapacitated. At such a time, any player can execute the inactive function and put the book into default, though there is no default fee because there is no way to charge the oracle. At this point, any taker can redeem their subcontract and withdraw their complete margin.

2.4 Gas Costs

The following are approximate gas costs for various functions

Create Book as new LP:	2400k, 2.8MM, $3
Take a position as investor:	220k, $0.25
Fund margin:	40k, $0.03
Withdraw from margin:	70k, $0.06
Withdraw from AssetSwap:	40k, $0.03
Cancel subcontract:	150k, $0.13
Settlement:	
fixed cost:	100k, $0.09
new takers:	20k per subcontract, 26k if default, $0.02
cancels:	25k per subcontract, $0.02
rollovers:	13k per subcontract, 15k if default, $0.01
final:	40k, $0.03

3 LP Risk and Return

The risk for an LP comes from her net exposure which will be randomly sided as LPs post two-sided offers at a singular forward price. This causes such net positions to be uncorrelated with any other asset. More formally, $\mathrm{Cov}(x, A\cdot x)=0$ if x is a random variable, and A is a random variable with equal probability of being +1 or -1. This independence simplifies the analysis because we need only anticipate volatility and not correlation in estimating portfolio risk.

For this example, we will assume the average funding rate is 8%, and the LP closing fee revenue is 3% of notional, for a total of 11%. Consider Alice, aka The Whale, has 100 ETH currently. We will assume the expected return for all assets is 0%, as we want to see the value of becoming an LP via its fees and target rate vs. its incremental risk, irrespective of any expectations about asset price movements. She contemplates allocating 25% of her ETH towards being an LP in the BTC contract. She expects her book to average a gross/net RM of 5:1.

Table 3

Current Portfolio and Portfolio with a 25% Allocation to OracleSwap

Current Portfolio	Portfolio with LP Allocation
100 ETH	75 ETH in Excess Margin
	25 ETH in Required Margin as BTC LP
	For example:
	75 ETH RM long/50 ETH RM short
	or
	50 ETH RM long/75 ETH RM short

The Whale's BTC position has a net RM of 25 ETH. This RM supports a net position long or short, but it also is subject to ETH value fluctuation as if it were still considered part of her initial ETH investment portfolio. The Whale's ETH risk is unchanged.

The net position it generates is a marginal risk. Given leverage of 2.5, an ETH price of $200, this 25 ETH supports $12,500 worth of a net long or short. Given the zero correlation of the randomly sided BTC exposure with her ETH position, we can ignore the covariance term in calculating her portfolio variance. That is, as the Whale can be either long or short, the raw correlation between her asset position and ETH is zero regardless of the correlation between that asset and ETH (even the case where the asset is ETH itself). Given an annualized volatility of 70% for the ETH and BTC, the USD volatility of these two positions are

Table 4

Portfolios with Annualized USD Volatility

Current Portfolio	Portfolio with LP Allocation
$20,000*0.7 or	$\sqrt{0.7^2 \times 20000^2 + 0.7^2 \times 12500^2}$
$14,000	$16,509

The Whale's total risk only rises by $2,509 by allocating 25% of her ETH to becoming an LP, much less than the $8,750 in risk generated by a $12,500 position considered by itself (her net BTC exposure).

Her 25 ETH supports 125 ETH in total book RM due to cross-margining (e.g., 75 long/50 short). Given a funding rate and closing fee revenue of 11%, a funding rate of 8%, and an extra 3% via her take of closing fees, this generates a $6,875 annual revenue. Her marginal Sharpe ratio is thus $6875/$2509=2.74. Note the Sharpe and returns are all invariant to the price of ETH or BTC, in that if the ETH price were initially $100 or $400, this would increase the initial and ending values by the same proportion.

The general formula for looking at an ETH whale's marginal risk is simply the difference between the portfolio with the allocation to the OracleSwap compared to her original ETH position:

$$\text{marginal volatility} = \sqrt{\sigma_{\text{ETH}}^2 + (\text{w} \cdot \text{LevRatio} \cdot \sigma_{\text{Asset}})^2} - \sigma_{\text{ETH}}$$

Here σ_{ETH} is the volatility of one's ETH holdings, and *w* the percent allocation of that to an LP contract *k*, which has a specific leverage ratio (LR_k) and volatility (σ_k). The initial ETH risk is unaffected by *w* because the net ETH position is unchanged by becoming an LP as some of her ETH has moved to the OracleSwap contract to act as margin and its USD fluctuation is the same as before. The diversification effect of idiosyncratic risk generates a nonlinear effect where allocating 25% of one's ETH to the OracleSwap generates less incremental volatility than when considered by itself.

The return will be a function of the Gross/Net margin, in that, an LP can have 20 ETH in their margin to support a book that is long/short various amounts: 20/0, 60/40, or 120/100. These generate the same amount of risk, yet the higher gross margins generate more revenue via the higher outstanding gross notional.

Table 4 shows how to compare a base long 100 ETH, to the case where The Whale allocates 25 ETH of that to be an LP. If you can understand this example, it generalizes to other scenarios via a simple closed-form solution.

Table 4

Assumptions		
ETH price in USD	$200	
Vol(ETH)=σ_E	70%	Annualized, as are all assumptions here
Vol(BTC)=σ_B	70%	
Correl(ETH,*z*·BTC)=0	0	*z* random +1/-1 with p=0.5
Leverage Ratio BTC contract (LR)	2.5	
Target rate on BTC contract	6%	
Portfolio A: base case, 100 ETH Long		
ETH quantity	100	Base long position
$Value	$20,000	ETH·ETHprice
$Vol	$14,000	$Value· σ_E
Portfolio B: allocate 25% to be LP in BTC contract		
% Allocation=w	25%	% of original ETH put into BTC contract as LP
Allocation in ETH	25	Amount of ETH in RM as an LP
Value(PortB)	$20,000	(750 excess margin + 250 RM)·ETHprice
Gross/Net	5:1	Anticipated ratio of Gross/Net: long 50/short 30, or long 30/short 50
Ξ Notional BTC exposure	62.5	Expected BTC exposure (in ETH) given net RM and LR. This is 'at risk' from BTC volatility.
$ Notional BTC exposure	$12,500	Notional BTC exposure in USD
$ Volatility of BTC Exposure	$ 8,750	$ BTC Notional·σ_B
$ Volatility of ETH Exposure	$ 14,000	ETH Risk does not change, as investor just moved some to act as margin
$ Volatility of BTC & ETH exposure	$ 16,509	Total portfolio volatility, BTC position uncorrelated due to random side
$ Marginal Volatility	$2,509	σ_{PortB} - σ_E
$ Marginal Revenue	$6,875	Gross/Net·net RM·LR·ETHPrice·Target Rate

Looking at the marginal Sharpe ratios for various contracts, with their specific LR and volatilities, we see the following Sharpe ratios as a function of the net RM as a percent of the LP's ETH portfolio.

Figure 3

Marginal Sharpe by Allocation to OracleSwap Contract

Net/Gross RM is 5. Funding rates were 8% for ETH and BTC, and 1.5% for SPX, while assumed fee income was 3% for cryptos and 1.2% for SPX. Volatility assumptions were 70%, for ETH and BTC, and 17.5% for the SPX, respectively.

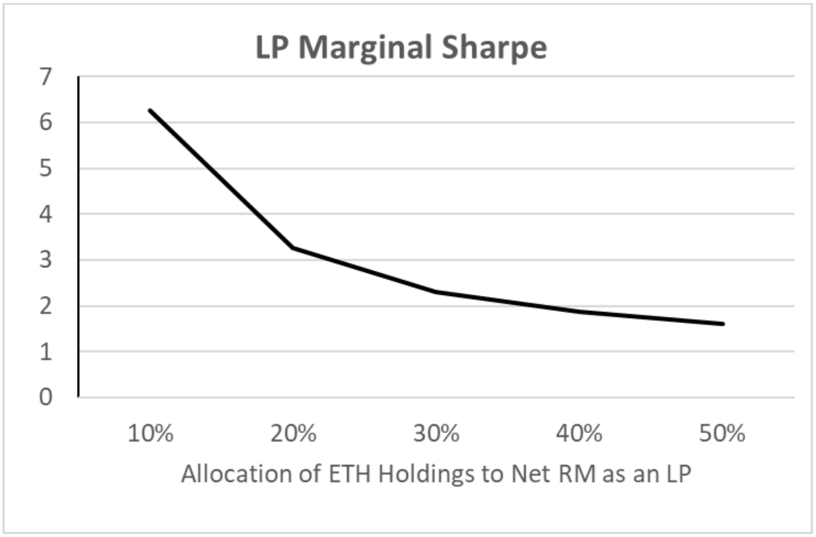


Figure 3 shows that for large ETH holders, the marginal attractiveness from a Sharpe perspective is exceptional for allocations below 30%, with Sharpe ratios well above 2. There are few real-world opportunities with Sharpe ratios above 1.[1] This example highlights that it is feasible to generate an attractive return for potential LPs while offering competitive rates if we target existing ETH holders who want to put that money to work while maintaining their custody and anonymity. These above-average returns should be feasible because, unlike in currency markets, it is difficult for large institutions to allocate large amounts of capital to arbitrage this given current regulatory constraints. Current large ETH holders have a comparative advantage in being LPs because of the low marginal risk of such positions.

[1] Historically, equity returns have a Sharpe ratio of around 0.4, and are considered a good long-term investment. Warren Buffet is a legend in equity markets, and his lifetime Sharpe ratio is around 0.64.

4 ETH Hedge Simulations

A perfect hedge can be achieved by putting the equivalent amount of ETH into margin as implied by the notional of the short ETH OracleSwap contract. For example, consider the following stochastic parameters for ETH in a binomial lattice:

Up Gross Return: 4/3 Down Gross Return: 3/4

The net returns are thus 33.33% and -25%, and with these returns, an up and down move lead to the original price. For an asset with a zero expected return, the probability of an up move, *p*, is simply

$$p = \frac{1 - down}{up - down} = \frac{1 - \frac{3}{4}}{\frac{4}{3} - \frac{3}{4}} = \frac{3}{7}$$

We want to have the probability of an up movement consistent with our expected return assumption, in this case, zero, as we wish to abstract from any assumptions about the expected return (it is trivial to add). That is, this probability ensures the expected return throughout the tree is zero at every node.

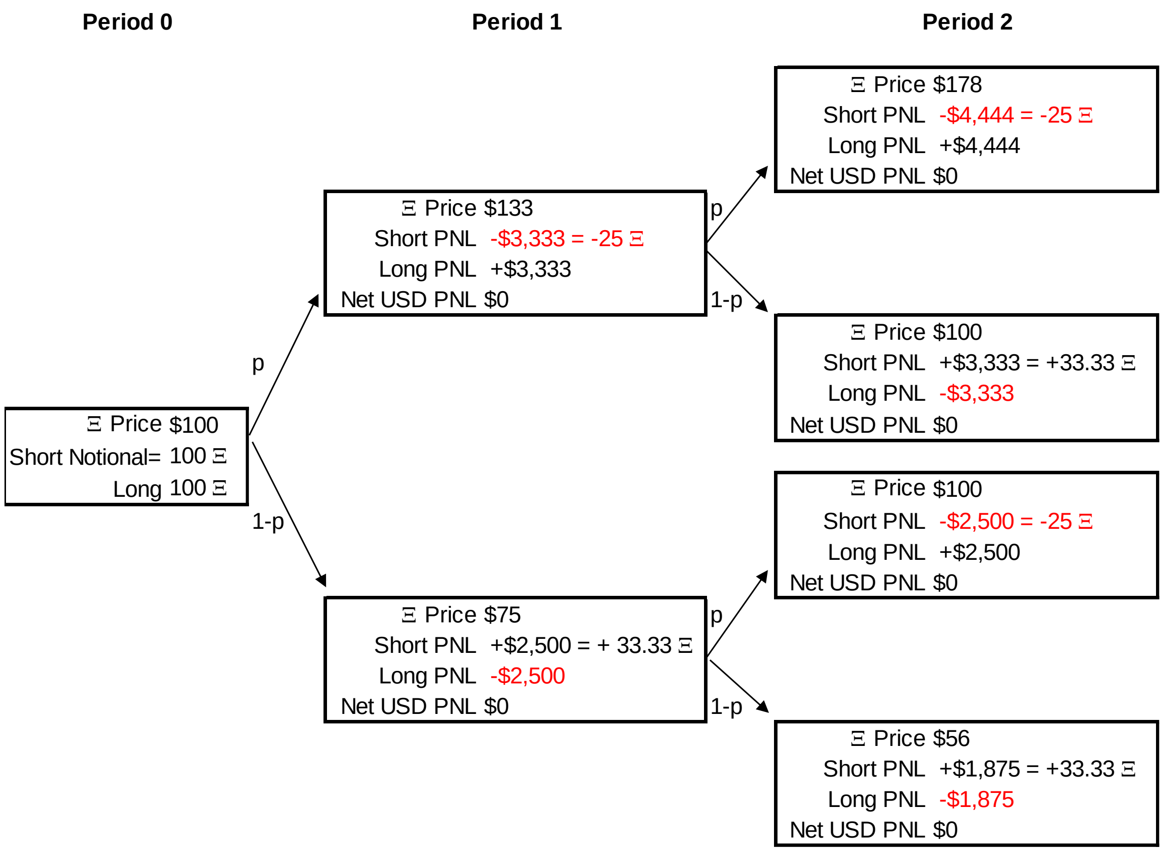
The shorthand for the profit, which is transacted in ETH, for USD denominated assets like ETH is:

$$PNL_t^A = Notional^{ETH} \cdot \frac{ret_A}{\left(1 + ret_{ETH}\right)}$$

This generates up and down ETH PNLs for the short of -25 and +33.33 ETH, respectively. This consistency is because the RM in this contract is fixed in ETH RM, so no matter how the ETH price moves, the ETH notional is constant. While the USD values of these cash flows will vary as the ETH price changes, the ETH PNL amounts are fixed. This generates the following path over 2 periods, where we note the PNL generated by the OracleSwap short contract with an RM=50, and notional value of 100. The ETH price starts at $100.

Figure 3

Hedged Returns with Constant Net Exposure

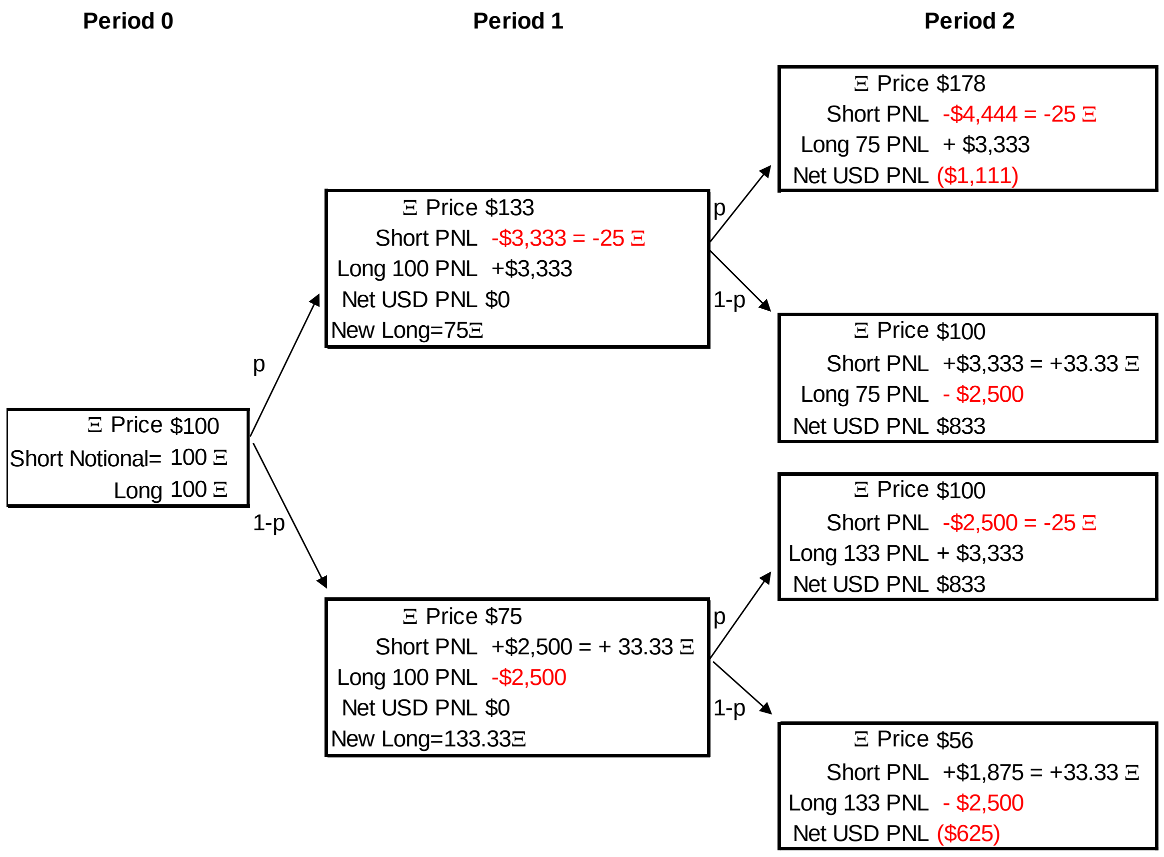


In the final period, all nodes also generate a $0 profit, reflecting a perfect hedge. However, to achieve this, one would have to buy or sell ETH in period 1. To reset one's margin to the initial margin is to 'renovate,' and to maintain a perfect hedge over time one has to renovate each time there is a cash flow, which in the case of OracleSwap is each week.

If instead, we left the ETH in the margin in period 1, we would have the following lattice:

Figure 4

Hedged Returns with No Net Margin Adjustments



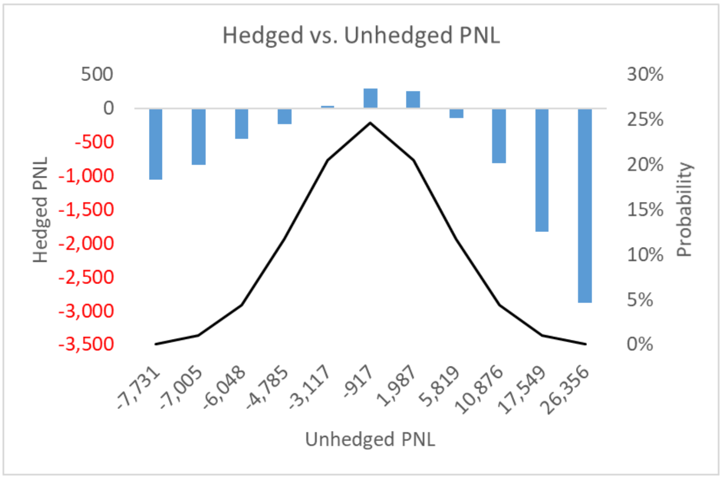
In this case, while the expected USD PNL is zero at every node, the final nodes have non-zero net USD PNL reflecting an imperfect hedge. This is because initially, the net position is long and short 100 ETH per the contract, but by period 1 there has been an ETH change to the net position due to the ETH PNL generated by the OracleSwap short position. If we leave our margin alone in period 1, then in the top node we would have a net short ETH position while the bottom node has a net long ETH position. Thus, in the subsequent period, the net position shows a profit if asset prices mean-revert, while there is a loss if the asset continues to rise or fall. This is why in simulations where the margin is not immediately set back to 100 ETH, the profit generated from a hedged portfolio (long 100 ETH, short 100 ETH notional in OracleSwap) creates a slight humped pattern to the final PNL distribution.

Figure 5 illustrates the outcome from a simple simulation using a lattice over 10 weeks. We calculate the total PNL after 10 periods for the unhedged and hedged position where the hedger applies a rule to cure his margin if it moves 35% from its original value. This generates an 85% reduction in total volatility over the 10 weeks, and on average requires only one cure.

Figure 5

Hedged vs. Unhedged ETH Portfolios over 10 Weeks

We generate all 1024 paths of ETH using a binomial approximation of a lognormally distribution with a mean return of 0 and an annualized volatility of 100%. The hedged USD PNL combines the long PNL with a PNL generated from an OracleSwap short of equal initial ETH notional value. The hedger then applies the rule of withdrawing all ETH above its initial ETH margin if it rises above or below 35% of its initial margin, so that an initial margin of 100 that rose to 137 would then be debited by 37 ETH and turned into USD; an initial margin that fell below 65 ETH, to 61 ETH, would take an infusion of 39 ETH, etc. Data are bucketed so each column represents an average of the hedged PNL



Data used in this example in OracleSwapData.xlsx available at www.oracleswap.co

5 The Oracle

5.1 Responsibilities

The oracle has the right and responsibility to post 4 PM prices to the oracle contract sometime between 4 and 5 PM, when users cannot take or cancel. Prices posted in the oracle contract are recorded in event logs and can be pulled outside of the EVM using web3.js. Contract constraints on the timing of these updates imply the Oracle can only post prices once a day, at most 5 days a week. Each Friday is settlement day, and when the settlement prices are posted, the Oracle contract computes that week's return from each day to the settlement price (up to 5 data points) and pushes it to the AssetSwap contracts.

Anyone auditing the oracle should be aware that historical crypto databases use different intraday closing times. The Winkdex index records 4 PM prices for ETH and BTC and so, currently at least, provides a useful reference for ETH and BTC prices. The median price from several exchanges, sampled at intervals that can vary up to 5 minutes, generates a standard error relative to a 'true' price. This variability would be fatal for a real-time system, but for long term investors, where neither side to a transaction has a latency advantage, such imprecision is both unbiased and immaterial.

The oracle protocol is to have several independent computers pull data from several independent sources that pertain to a few-minute window. Given more than five sources, the median is a robust estimate of that day's closing price. For assets as liquid as BTC and ETH a strategic attempt to move world prices significantly for several minutes would take several orders of magnitude more resources than the bZx exploit. If server 1 fails to pull or post prices, then server 2, seeing the Ethereum contract's last update time is stale, tries to post prices it pulled from a time window a few minutes later in case the problem was related to the time of day. A third computer checks again.

5.2 Attack Surface

A fraudulent price is the only attack surface in this protocol. Outside of margin funding and withdrawal (which can only be done by the account owner), or a default penalty attributed to a taker at redemption, margins are only debited and credited at the weekly LP book settlement via the returns created by the oracle contract. The algebra used to generate ETH PNL is coded into the oracle and book contracts and cannot be altered.

As the only cheating scenario is the Oracle posting fraudulent prices that inflate the PNL of a conspiring counterparty, we will call this combo oracle and counterparty Oracle/Alice (in practice, the Oracle would probably be Alice to avoid diluting his payoff). OracleSwap's incentive compatibility comes out of the way it is a repeated game. Given the oracle revenue via close fees, the oracle owns an annuity that is non-transferable. A single cheat should be fatal to all contracts the OracleSwap's oracle services, like the grim trigger strategy in a repeated game. We can then compare the value of this annuity to any cheat payoff the Oracle may generate.

In iterated prisoner's dilemma games, the optimal strategy is not to play the Nash strategy of the 1-period game, but to cooperate and play a socially optimal strategy. The value of a repeated game is that uncooperative play (aka cheating, defecting) reduces the payoff to both players in future periods. A player may choose to act selfishly to increase their own reward rather than play the socially optimal strategy, but if it is known that the other player is following a trigger strategy (never cooperate again once a counterparty defects), then the player expects to receive reduced payoffs in the future if they deviate at this stage. An effective trigger strategy ensures that cooperating has more utility to the player than acting selfishly now and facing the other player's punishment in the future. This is *reciprocal altruism*: I play nice because I expect you will respond by playing nice too.

A key to this game is that players can easily and accurately see if an agent is cheating, highlighting the importance of an oracle that is easy to audit. While everything is immutably recorded on the blockchain, many oracles are difficult to review. This lack of transparency not only makes it difficult for new users to trust an oracle, but it also provides the oracle time to exit before outsiders can punish him. OracleSwap makes it easy to see its current and historical prices.

5.3 Common Knowledge and The Burn

If Oracle/Alice has net positions across several different subcontracts, they could post prices such that the net position generates the maximum PNL. For example, if net long (whether as an investor or LP), they would post a price high enough such that their long PNL=RM. The burn option encourages the cheated counterparties to prevent Oracle/Alice from capturing this payoff.

In the *settlement period*—between the Oracle contract settlement update and LP book settlement—players can do any of the following for contracts they have not already canceled.

- add to their margin to prevent a default
- burn their debit rather than let their counterparty enjoy it
- do nothing
    - implying *default* if total margin + prospective PNL < RM
    - implying *continue* else

Counterparties cannot withdraw or cancel during the settlement period. The cost of default and burn is the same, RM/2.

Assume Bob is the counterparty to Oracle/Alice. Let us define PNL^{true} as the true PNL given the subcontract parameters and honest prices, PNL^{rep} as the PNL derived from the reported prices which may be untrue, where PNL is from Oracle/Alice's perspective (Bob's cash flow is thus -PNL). An honest oracle posts prices that imply $PNL^{rep}=PNL^{true}$.

If Oracle/Alice cheat at the settlement via a bogus price report, they have no downside from further exposing their cheating nature the following week: outsiders will see the cheat in our Oracle Price Contract's event logs, and rational investors will not risk their money with a cheating Oracle. This implies Oracle/Alice should post prices implying $PNL^{rep} \geq \max(0, PNL^{true})$ at the next and presumably final period because there is nothing Bob could do to further damage Oracle/Alice's reputation if they post $PNL^{rep}=0$ when $PNL^{true} <0$, and as Oracle/Alice would owe nothing to Bob, his burn would not cost Oracle/Alice anything. If $PNL^{true} >0$ in the final period and they report truthfully, presumably, Bob will accept this report as well because in that case burning would not punish the Oracle (because the Oracle/Alice is paying). Thus, Bob should rationally expect a cost of at least $E[\max(0, PNL^{true})]$ for playing another week, which is about 0.4σ (where σ is a subcontract's 1-standard deviation PNL for a week).[2]

Oracle/Alice then reason they should expect to get away with charging Bob $\max(0, PNL^{true})+0.4\sigma$ in the next and presumably final period because when Bob was presented the choice of implicitly paying an extra 0.4σ vs. the burn cost, he made the cheaper choice (a burn costs RM/2 or about 1.5σ). This is based on the idea that rational decision making ignores sunk costs. A cheating Oracle has cost Bob not just (PNL^{true} - PNL^{rep}), the amount of the initial cheat, but also the implicit 0.4σ cost, the amount of the expected value of the anticipated cheat.

Thus, if Oracle/Alice decide to confront Bob with the choice of paying an additional 0.4σ or burn in the next and final settlement, we should expect the same choice to not burn, the same decision Bob made previously (being rationally indifferent to sunk costs). Bob anticipates Oracle/Alice reasoning in this

[2] If x a standard normal, $E(y| y = x \text{ if } x > 0) = ½\cdot E(x| x > 0) + ½\cdot(0|x < 0)=E(x| x > 0) = \text{sqrt}(2/\pi) \cong 0.4$.

fashion, and thus now expects Oracle/Alice to report max(0, PNL^{true})+0.4σ, which has an expected value of 0.8σ.

Oracle/Alice anticipate this level of Bob's reasoning, and so using similar logic as before, assume if Bob does not burn when expecting a cost of 0.8σ, this implies Oracle/Alice can over-charge Bob by 0.8σ in the final period and not arouse Bob's burn response. Bob anticipates this reasoning, and now his expected cost in the next and final period is max(0, PNL^{true})+0.8σ, which has an expected value of 1.2σ.

If Bob chooses not to burn, Oracle/Alice would then anticipate they could over-charge Bob by 1.2σ in the final period and not arouse Bob's burn response, for an expected value of 1.6σ.

Given the cost of a burn is RM/2, which is 1.5σ, at this point, Bob realizes that if he does not burn, Oracle/Alice's final price report will cheat Bob by more than the cost of burning and immediately terminating the subcontract.

For players with Excess Margin<RM/2, defaulting will be cheaper. Considering most MakerDAO contracts are overcollateralized by 300%, and that over-collateralization is an efficient way to manage a margin position, it is reasonable to assume most counterparties will probably have more Excess Margin greater than RM/2, i.e., the burn cost. All such players rationally should burn rather than default or continue.

Rational analysis nicely aligns with our less-rational instincts here. *Altruistic punishment* describes how people punish non-cooperators at a cost to themselves even in one-shot interactions where there is no chance of any long-run benefit. Experimental and ethnographic data shows that altruistic punishment strengthens cooperation.[3] People hate cheaters and will pay to hurt them. For example, in games like *tit-for-tat* or *the ultimatum game*, people cooperate more when there is an option for players to pay to punish players in the game who are playing a rational but socially uncooperative strategy. As the oracle receives the default fee, burning instead of defaulting has the additional benefit of preventing a cheating oracle from profiting here. Potential cheaters rationally anticipate this individually irrational punishment, discouraging cheating.

5.4 Burn Mechanics

If a taker burns, the effect works as follows. Each burn reduces the 'burnFactor' that is multiplied against all PNL credits. That is, only those receiving ETH are affected by burns. The default burnFactor is 1.0, so players get 100% of their PNL in the standard case. If a long taker representing 10% of the long book burns, the burnFactor is reduced by 10%. If another taker representing 12% of the long book burns, this reduces it by another 12%, down to 78%.

If the LP burns, their effect is calculated using the LP's net position. For example, if the LP is long 10 and short 6, she is net long 4, and so if she burned it would reduce the burnFactor by 40%. She would have to pay her RM/2 just like a taker, but her RM is on the net position, not the gross. Similarly, the LP's net

[3] Fehr, Ernst, and Gachter. “Cooperation and punishment in public goods experiments.” *American Economic Review* (2000). The innate human desire to apply costly vengeance paradoxically lowers violence as these hard-wired emotions act like commitment devices that discourage opportunistic, rational, but socially destructive acts, and is a crucial motivator for cooperative behavior. In contrast, in many species like lions and baboons, a new male alpha rationally kills the unweaned babies because females do not hold irrational grudges about sunk costs, and so soon become receptive mates for the new alpha, creating an incentive for the next alpha to kill the infants.

position is used for attributing the burn. That is, her net credit would be adjusted by the burn factor, not the gross credit.

An evil oracle would rationally cheat only at the settlement price. Cheating intraweek merely gives people more time to cancel and burn, reducing their potential heist. A cheat price would set an asset price either very high or low, cheating all the longs or the shorts, but not both. While the longs *and* shorts could all burn, this means they would be paying money to receive less. This would be perverse, but even if someone did this, the burnFactor is bounded below by zero.

The burn reduces the credited amount to the greater of the LP's long and short sides. This keeps the burn factor from going below zero. A cheat would probably involve the large of the two sides, because otherwise it implies the cheating oracle left money on the table. This does not matter much to incentives for two reasons. First, the incentives for oracle honesty exist without the burn. Secondly, while it diminishes the burn penalty (dividing by 10 instead of 5), a cheat from the smaller 'leg' implies a greater amount of positions unaffiliated with the cheat. Thus the forgone revenue is proportionately higher.

5.5 Total Cost-Benefit to Cheating

Let us benchmark our exit scam payoff via the Total OracleSwap RM, which is the sum of all the RM across all the OracleSwap subcontracts. We will denote this RM_{gross}. A contract attractive to hack will have many users, and many will be accidentally aligned with the cheating Oracle's position side. A cheating oracle that tries to build a position slowly would have to expose themselves to many random PNLs, and as their maximum payoff is their net position's RM, this is a 3-standard deviation weekly move, potentially losing money via market moves could create a scenario where the cheater's benefit—outside the loss of future oracle revenue—is negative. This implies a cheat would most likely entail a quick positioning, which implies the cheater's positions would be a minority of the positions within the AssetSwap contract. These independent positions, those not related to the oracle's cheating strategy, cut down the percent of RM_{gross} affected by an Oracle scam by at least ¾, in that the cheating oracle need to put up positions quickly to avoid exposing himself to too much risk (this is, after all, the benefit of cheating ignoring the costs). A book big enough to be worthy of stealing from would not digest a large demand/supply shock. When combined with the fact that most players will burn rather than continue, say ¾, this cuts Oracle/Alice's final payoff to $\frac{1}{4}\cdot(1-\frac{3}{4})$ or $\frac{1}{16}\cdot RM_{gross}$.

If we assume subcontracts roll over every 2 months, then the Oracle expects 6 closes, and as each subcontract generates 2.5% RM in Oracle revenue, the annual dividend on RM_{gross} is 15%. Using the Gordon dividend discount model, a discount rate of 10% and a growth rate of 5% imply a present value of 3 to $6\cdot RM_{gross}$. Thus, under very modest assumptions, an evil oracle would find it rational to use their Oracle revenue to further other evil schemes, because even evil people prefer more money to less ($3 > \frac{1}{16}$).

Higher expected growth rates make the value of the annuity lost even greater, making cheating less attractive. The key is that unlike many other Oracle business models, even upon achieving a steady-state level of growth, the Oracle finds honesty the dominant strategy. A seasoned OracleSwap oracle will have perfected the automated scripts that attend to its contracts, so its annuity stream would take virtually no effort or capital, an asset worth preserving. OracleSwap is a long-run incentive compatible mechanism.

5.6 The Value of the Walk-Away Option

Consider the strategy of playing until one generates a loss and then posting a fraudulent price. While the counterparty would probably burn, the cheating Oracle would have lost nothing.

There is a trade-off in such a strategy as higher returns imply higher risk. For example, the strategy of walking away from a substantial loss, say a -2.5σ outcome would take an average of 161 weeks for this to happen. While the expected cumulative payoff is 2.8σ of a weekly standard deviation, its standard deviation is 12.7σ, meaning their expected return would be between 15.5σ and -9.9σ, a poor Sharpe ratio and even worse in terms of time wasted. A rational cheating oracle has to balance the expected return to the time and volatility of such a strategy.

Looking at the cheat strategy payoff strategically is the classic *optimal stopping problem*. Specifically, Oracle/Alice's expected payoff is of the following form.

$$\text{Cheat Payoff} = \sum_{t+1}^{T-1} PNL_t \; where\, T = min\; t\; s.t. PNL_t < k$$

Diabolical Oracle/Alice's strategy is to post true prices until they imply that $PNL^{true} \leq k$, at which time they will report prices that imply their PNL=RM, Bob would burn, terminating the contract, and Oracle/Alice withdraw their margin ETH and move to Belize to work on their next white paper. The sum of PNLs up to time *T* has a positive expected value because the sequence of PNLs from *1* to *T-1* are all random variables truncated from below. For example, if *k*=0, the only PNL that will accrue to Oracle/Alice will be positive; they will walk away from their first loss.

While the expected payoff from such a strategy increases linearly as *k* moves from 0 to $-\infty$, the volatility increases exponentially. Oracle/Alice's problem is one of maximizing a risk-adjusted return.

The ratio of the expected payoff to the standard deviation is maximized at around *k*= -0.5, though the maximum is exceptionally flat from 0 to -0.5. This implies Oracle/Alice's Sharpe ratio would be 90% near optimal, and a lot quicker and simpler if she used the strategy to walk away from her first loss. The expected total return to the optimal 'walk away from one's first loss' strategy is about 1σ, which is about ⅓RM.

Other counterparties not part of the conspiracy would reduce this by another ¾ in comparison to the RM_{gross} used to generate Oracle fee revenue. This reduces the stopping strategy payoff to ¼·⅓ RM_{gross} or $\frac{1}{12}\cdot$ RM_{gross}.

The problem with this scam is it is improbable that one contract will break a target loss the same week another one does. As no one should trust an oracle that cheated on a different contract, this means the oracle reaps the returns from one contract at the expense of all the rest, making the cost-benefit much worse. For example, if the SPX loses one weekly standard deviation, Oracle/Alice's other AssetSwap positions (say, on the ETH) will, on average, be near zero, generating little marginal value to this scam. As the number of contracts increases, the ratio of this exit scam payoff to the RM_{gross} declines.

An ambitious Oracle has little to do but think of new contracts of interest, so the addition of new assets is a reasonable Oracle expectation, implying a large expected growth rate as the opportunity cost of cheating (loses that potential revenue). An oracle with many assets, meanwhile, would only be able to do this for a small fraction of the assets serviced (those that are highly correlated), however, unlike the more deliberate cheat where the oracle can implement a targeted cheat across all assets simultaneously. For this reason, it is irrelevant as a practical consideration.

7 Contract Functions

The explanations refer to ETH, which is 1e18 wei. The contract running at oracleswap.co, that is downloadable at GitHub, uses Szabo as the unit of denomination, which is 1e12 wei. This is so that I can present the functionality while not violating any US regulations, as these amounts imply economically meaningless payoffs to all involved. Anyone building upon the code provided, in Solidity and Javascript, should adjust constants upward by 1e6 to make it relevant.

7.1 Take position

1 transaction

1. Function: take(LP address, RM, side)
    a. Find LP
    b. Set RM, which sets the notional (2.5 x for ETH and BTC, 10x for SPX)
    c. Set side, true for taker long/LP short, false for taker short/LP long
    d. Send 1.5 x RM with function to initialize margin

Example:

Input: 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 2, true

Must send at least 3 ETH.

In the video example, the taker sends 5 ETH, and signifies they are long by toggling the 'long' radio button.

This take a long position (long == true), with an RM of 2 ETH, with LP 0xCA3...33c

The investor's position is identified by the LP and the subcontract ID generated; {LP address, SubkID}

7.2 Taker Curing Margin

1 transaction

1. send eth to margin
    a. function for taker: takerFund(LP address, subkcontract ID)

The settlement price update on Friday at 4 sends the ending price for that week's settlement. Users can see their impending ETH debit or credit to see if they need to cure their margin and avoid a default. For the investors, this is easy: add the impending PNL to one's current margin, and if it is below the required margin, they will default at settlemnet if they do not cancel or cure. Cancelling can be done anytime prior to the Friday settlement price update. After that, the only option is to cure their margin position by sending enough ETH so that their total margin, after the PNL debit, is greater than their RM. They have until at least 24 hours after the oracle settlement price update to send ETH to their margin.

7.3 Taker Cancel

3 transactions (including withdrawals)

Cancels, like takes, are irreversible. They require a fee which is the sum of the oracle and LP closing fee. For the LP, a cancel is twice the oracle closing fee.

There are two types of cancels. A regular cancel uses the weekly settlement price as its last price point, and any regular cancel can be implemented at any time prior to the oracle settlement price update. This involves setting the bool in the cancel function to false.

If one is very nervous and wishes to get the very next oracle price as their final price, they can toggle the cancel bool to true. Canceling intraweek generates complications for the LP, in that the LP could then be exposed to more market risk, making their current net RM inadequate. To prevent this situation, we only allow the investor to close intraweek if the LP has sufficient excess margin available. For example, if the LP has a flat book and zero margin, investors cannot cancel intraweek prior to the next settlement day; investors in such a scenario could only put in a cancel that would go into effect at settlement using settlement day prices. As an LP should have significant excess ETH—LP economics make the most sense for existing ETH whales—most LPs should have excess margin. The investor pays the maximum LP closing fee to compensate for the impact of the intra-week close on the LPs margin. This gives LP an incentive to maintain the excess margin that allows these intraweek cancels.

The intaweek cancel is still settled at the weekly settlement, so taker must wait until then to retrieve their funds. This intra-week settlement option is only available to the taker, not the LP. An intraweek cancel implemented on the last day of the week is treated like a regular cancel.

After a cancel, the money in an investor's margin cannot be used for anything else, such as transferring it directly to another subcontract, and so should be withdrawn as soon as possible. A cancel, burn, or default terminates a subcontract at the settlement. The investor's margin on an inactive subcontract can only be withdrawn via the 'redeem' function, which moves the investor's margin to the AssetSwap contract and pops the subcontract from the list of the LP's subcontracts. After that, the investor should withdraw their ETH from the AssetSwap contract. If the investor defaulted on an active LP book, they are debited Max(takerMargin, 0.5·RM) at redemption, which is sent to the oracle.

As an inactive contract takes up space in the LP's limited array of taker contracts, the LP, like anyone else, can redeem an inactive contract. Yet if the LP redeems the inactive subcontract, she receives 2 finney from the investor's margin to compensate for gas and effort.

Taker Cancellation steps

1. Cancel before Friday oracle update (or Thursday if Friday is a holiday)
    a. Cancel(LP address, subkID bytes, now bool)
    b. Wait for LP book settlement
2. Redeem
    a. Redeem(LP address, subcontract ID)
    b. Moves ETH to AssetSwap contract
3. WithdrawFromAssetSwap
    a. withdrawFromAssetSwap()

7.4 Inactive LP

1 transaction

1. inactiveLP(LP address)

If the LP is inactive, in that 48 hours passed since the oracle settlement without an LP settlement, the first taker within the LP's book to execute the LP inactive function gets the LP's default fee of max(totalLPMargin, LP RM/2). This then inactivates the LP book, allowing all takers to costlessly redeem their subcontracts immediately, and for the LP to withdraw all her remaining margin.

7.5 Become an LP

1 transaction

1. createBook(RM, close fee, long Funding rate, short funding rate)
    a. Set minimum RM, closing fee, long funding rate, short funding rate
        i. Data are in basis points as a percent of notional
        ii. Max close fee for the LP is 100 for SPX, 25 for ETH/BTC
        iii. Min/Max funding rates are +/- 100 for SPX, +/- 25 for ETH/BTC
    b. Send 10x minimum RM to initialize margin

Example input: {3, 10, 20, -10}

He must send at least 30 ETH

This creates a book with a minimum RM of 3 ETH, a closing fee for the LP of 10 basis points as a percent of notional, a 20 bp funding rate for long investors, -10 bp funding rate for short investors.

The LP's book is identified by the LP's eth address. A payment of 0.05 ETH is required at the time of book creation. While this fee goes to the oracle, the purpose is to prevent someone from trying to clog the system creating many accounts they do not intend to use.

7.6 LP Curing Margin

1 transaction

2. send eth to margin
    a. function for LP: lpFund(LP address)

The settlement price update on Friday at 4 sends the ending price for that week's settlement. Users can see their impending ETH debit or credit to see if they need to cure their margin and avoid a default. For LPs estimating their cure can be a little complicated. Not only do they have an impending PNL, but cancellations at settlement may increase their RM so that even if they have a positive PNL, they still need to cure their margin. Also, if an investor defaults that subcontract is removed from their book, and so this too may cause their RM to increase. New positions are prevented between the Friday oracle price update and the book settlement, so the LPs should have a good idea of which subcontracts are cancelled, and which are potentially going into default. Defaults are costly and avoidable, so hopefully users will be smart enough to make them rare. Initially, an LP should be prepared for such a scenario, and over time develop a better sense of how frequently this occurs.

7.7 Withdrawing margin

2 transactions

Withdraw functions move ETH from user margins in the book contract to the AssetSwap contract. This cannot be done for takers with inactive subcontracts, which instead must use the 'redeem' function that deletes the subcontract from the LP's struct, debits any default fee if applicable, and then sends the entire margin to the investor. This also cannot be done during the settlement period between the weekly oracle settlement price update and the LP's book settlement.

1. Withdraw from margin
    a. LP: withdrawLP(amount)
    b. Taker withdrawTaker(amount, address, subcontract ID)
2. Withdraw from AssetSwap
    a. Both: withdrawFromAssetSwap()

7.8 LP cancellation

1 or 2 transactions

There are two cancellation methods for LPs. The first is a standard cancellation of a subcontract, which can only be closed at the settlement price, not intraweek. The LP does not redeem the subcontract, rather, at settlement the RM is adjusted for the LP to reflect the cancellation and the LP can withdraw their excess margin.

If an LP has a large book and wants to close the book, cancelling each specific subcontract would be onerous. Therefore, we allow the LP to exit en masse, though it requires a wait time of 28 days. The charge is 1.0%/0.25% of the larger of the LP's long or short positions. This fee is charged at both steps in he book cancel function. This is cheaper than canceling each trade separately (which would apply a charge of 2.0%/0.5% to the sum of the long and short positions.

To enact this, the LP must first execute the function endBook, which sets the cancellation at 28 days after its invocation. This cannot be undone. Once this day has passed, the LP must execute the activateEndBook function, which tells the LP book to allow all takers to redeem their subcontracts at the subsequent settlement, and to never process a new take or settlement. The next settlement will be the last settlement for the LP's book, which also will be unable to take new positions. After the final settlement, the LP can withdraw their entire margin to the AssetSwap, and then from the AssetSwap to their personal account.

Regular cancel, 1 step

1. Cancel before Friday oracle update (or Thursday if Friday is a holiday)
    a. cancel(address, bytes32, bool)
        i. LP can only set bool to false because they cannot cancel intraweek.
    b. Wait for settlement, see RM adjusted appropriately

Book cancel, 2 steps

1. Set book closing time 28 days from day activated
    a. cancelBook()
2. After 28 days, run function again to proceed with cancellation of all subcontracts at the next settlement. checks to see if today is after the book closing time. If true, LP can activate, which closes the book at settlement. At that point all takers can costlessly redeem their subcontracts, and the LP can withdraw their entire margin

   a. cancelBook()

7.9 LP Settlement Execution

1, or 4+ transactions

Anyone can run the settlement function, as it can only be run once each week, and as it costs gas, the LP should be pleased to let someone else pay to do this. This could also be useful if an evil LP tries to avoid a PNL debit by not settling or settling part of their book, in that someone else could settle all or part of the book for her.

Settlement function is designed for scale, allowing LPs to have up to 4000 subcontracts. Processing these efficiently requires breaking up the subcontract settlement into 3 different cases: new, expiring, and roll-overs. It also then processes the final LP settlement—defining her new RM, updating her margin for that week's cumulative PNL credit or debit. For LPs with less than 200 subcontracts, the settleBatch function should be used because it will work and cost less gas. For LP's with more than 200 subcontracts, the LP should run the 'settleParts' function until the settlement is completed. This will then take at least 4 steps. It will take more than 4 if one of the categories: new, total, expiring, contains more than 200 subcontracts.

For small books (< 200 subcontracts)

- Run settleBatch

For large books

As there are 4 functions in the settlement, this means settleParts must be executed at least 4 times.

1. Settle expiring contracts
   a. Requires oracle settlement price update > last book settlement
      i. This prevents multiple book settlements
   b. Requires 24 hours after oracle settlement price update
   c. Requires settleNum < 1e4
   d. Program loop capped at 200
   e. repeat if there are more than 200 expiring contracts
   f. when completed, settleNum = 1e4
2. Settle rollovers
   a. Requires settleNum < 2e4
   b. Repeat if total # positions > 200
   c. When completed settleNum = 2e4
3. Settle new contracts
   a. Requires settleNum < 3e4
   b. Repeat if total # new positions > 200
   c. When completed, settleNum = 3e4
4. Settle LP
   a. Requires settleNum = 3e4
   b. When completed, settleNum = 0, book settlement time updated

The LP book contract variable, settleNum, monitors its progress through LP's subcontract sequence and makes sure that settlements are completed. Given gas constraints, settlement only loops through at most 200 subcontracts per execution. This means that if an LP has 402 expiring subcontracts the second step must be executed three times. It will show settleNum = 200 after the first settle, 400 after the second, and 10000 after the third. Given the settleNum==10000, the rollover positions can be settled in a similar way. When the rollovers are complete, settleNum == 2e4, which allows the new positions to be settled, after which settleNum= 3e4, allowing the LPs margin to be settled. At the end of this process, settleNum = 0, and as the book settlement time is updated and now after the oracle settlement time, settlement functions canont be executed until the next oracle settlement update.

7.10 LP Close Fee and Funding Rate Adjustments

1 transaction

1. updateFees(uint newClose, int frLong, int frShort)

LP's can adjust their fees at any time. They input the data in basis points so 10 is 0.1%. It is the fee in terms of the notional, so when applied in the settlement or closingfee, it is multiplied by the leverage ratio to get the factor that is applied to the RM.

	Close Fee		Funding Rate	
	ETH or BTC	SPX	ETH or BTC	SPX
Min	0	0	-100	-25
Max	100	25	100	25

7.11 LP Adjust minimum RM size

1 transaction

1. adjustMinRM(min amount)

The minimum RM for an LP can be adjusted to avoid smaller subcontracts, but also is a way to avoid new takes, in that if set to the maximum (65535), there will be no more new takers if one does not have enough excess margin to handle such a new subcontract.

7.12 InactiveOracle

1 transaction

1. inactiveOracle(LP address)

If the oracle is inactive and has not posted new settlement prices for 10 days, anyone can execute the inactiveOracle funciton, which allows all takers to redeem their subcontracts immediately, and the LP to withdraw its entire margin. Once marked inactive, no new takes or settlements can occur.

7.13 Payable functions

For functions cancel and burn, a payment for a fee must be sent. Any excess payment over the amount owed via contract parameters is sent to the payer's margin.

- burnTaker(LP address, subcontract ID)
- cancel(LP address, subcontract ID, bool)
- createBook(RM, close fee, long Funding rate, short funding rate)
- fundLP(LP address)
- take(LP address, uint256, bool)
- fundTaker(LP address, subcontract ID)
- burnLP(LP address, subcontract ID)

7.14 Nonpayable functions that require gas

- inactiveLP(LP address, subcontract ID)
- inactiveOracle(LP address)
- redeem(LP address, subcontract ID)
- settleParts(LP address)
- settleBatch(LP address)
- withdrawFromAssetSwap()
- withdrawLP(uint256)
- withdrawTaker(uint256, LP address, subcontract ID)

7.15 Getter functions (no gas fee)

- getBookData(LP address)
    - input: LP address
    - book (LP address)
    - lpMargin (in wei)
    - totalLpLong (in wei)
    - totalLpShort (in wei)
    - lpRM (in wei)
    - bookMinimum
    - longFundingRate
    - shortFundingRate
    - bookStatus

        0: fine
        1: cancelled, all contracts expire at next book settlement
        2: LP burned, all subcontracts expire at next book settlement
        3: Book dead via burn, default, cancel, everyone can redeem asap
    - currentCloseFee
- getSettleInfo(LP address)
    - totalLength: number of total subcontracts in LP's struct list
    - expiringLength: number of expiring subcontracts
    - newLength: number of new subcontracts
    - lastBookSettleTime
    - SettleNumber
    - bookBalance
    - bookMaturityUTC
- getSubcontractData1(LP address, subcontract ID)

    - taker address
    - tarker margin
    - required margin
- getSubcontractData2
    - subkStatus

        0: new contract
        1: active, neither new or canceled.
        2: taker has canceled, subcontract expires at next book settlement
        3: LP has canceled, subcontract expires at next book settlement
        4: investor has canceled intraweek, though subcontract expires at next book settlement
        5: taker has burned, subcontract expires at next book settlement
        6: taker defaulted at settlement and will be debited 50% of their RM at redemption (contract will burn whatever is there if taker margin is less than 50% of RM).
        7: subcontract is no longer being settled, but it still exists in LP's book. Taker should redeem to withdraw margin to asset Swap contract, then withdraw from the AssetSwap contract to their personal address
    - priceDay
    - closeFee
    - fundingRate[2]
        - 0 for short funding rate, 1 for long
    - takerSide
- hourOfDay()

7.16 User Rights and Restrictions

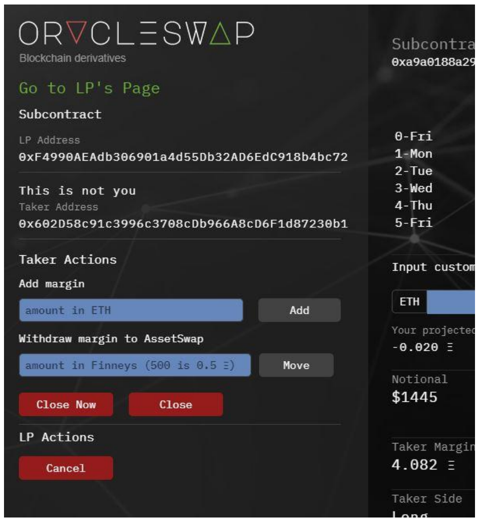
- No takes between 4-5 PM
- Settlement Period (0 to 24 hours after Oracle settlement price update) restrictions
    - Approximately Friday 4 PM through Saturday 4 PM
        - No new positions
        - Users cannot withdraw margin
        - Users cannot cancel
- Users can fund at any time
- Users can withdraw from the AssetSwap contract at any time
- Users can create new books, become an LP, at any time
- LPs can change fees at any time
    - Subcontracts use the fees posted at time of instantiation for the life of the position
- Users cannot withdraw below their RM while subcontract is active
    - Subtracting the amount requested for withdrawal must result in a margin greater than or equal to the required margin. After default, cancellation, etc., takers can withdraw via the redeem function, while the LP can withdraw via her withdrawLP function.
- Users can withdraw if LP misses settlement
    - If the LP's book has not been settled within 48 hours of the last oracle settlement price update, the first taker to execute the inactive LP function receives the LP's default payment, and then everyone can withdraw their entire margin via the redeem function. The book cannot take new subcontracts or settle again.
- Users can withdraw if oracle inactive

- If the book has not been settled for 10 days, this means either the LP is inactive or the oracle is inactive. As users have an incentive to invoke the inactiveLP function, the only reason a book can continue for 10 days without a settlement would be if the oracle has not posted new settlement prices. Thus, we only need this one restriction to capture the root problem. There is no direct benefit for flagging an inactive oracle, but once done, all players can withdraw their margins.

7.17 Conditional Web Rendering

As there are several restrictions, such as that users cannot cancel during the settlement period, the web front end has conditional rendering that removes infeasible options to minimizes mistakes. If these function calls were made they would be rejected, and no gas fee would apply, but it is straightforward to add a condition on the webpage removing any potential for such wasted time.

In normal times an investor can withdraw and add margin, and also close.



Subcontract

LP Address
0xF4990AEAdb306901a4d55Db32AD6EdC918b4bc72

This is not you
Taker Address
0x602D58c91c3996c3708cDb966A8cD6F1d87230b1

Taker Actions
Add margin
Withdraw margin to AssetSwap

LP Actions

Subcontra
0xa9a0188a29

0-Fri
1-Mon
2-Tue
3-Wed
4-Thu
5-Fri

Input custom

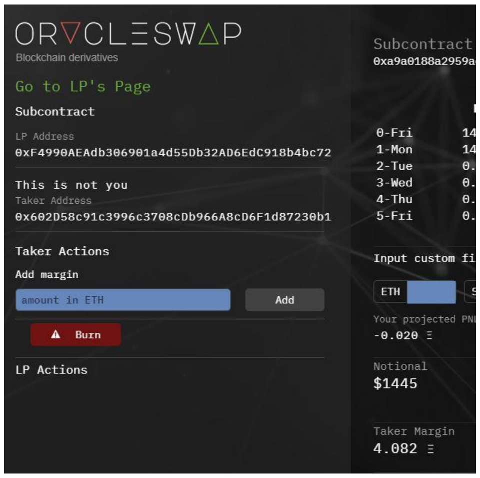
Your projected
-0.020 Ξ

Notional
$1445

Taker Margin
4.082 Ξ

Taker Side

During the settlement period the investor can only add margin, or burn.



Subcontract

LP Address
0xF4990AEAdb306901a4d55Db32AD6EdC918b4bc72

This is not you
Taker Address
0x602D58c91c3996c3708cDb966A8cD6F1d87230b1

Taker Actions
Add margin

LP Actions

Subcontract
0xa9a0188a2959a

0-Fri 14
1-Mon 14
2-Tue 0.
3-Wed 0.
4-Thu 0.
5-Fri 0.

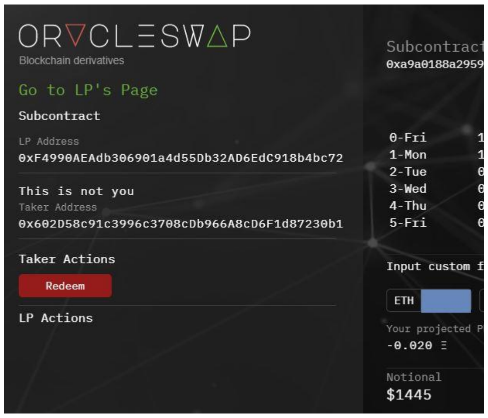
Input custom fi

Your projected PN
-0.020 Ξ

Notional
$1445

Taker Margin
4.082 Ξ

When position is inactive, such as after its final settlement after a cancel, the user can only redeem their margin.


ORVCLΞSWAP
Blockchain derivatives

Go to LP's Page

Subcontract

LP Address
0xF4990AEAdb306901a4d55Db32AD6EdC918b4bc72

This is not you
Taker Address
0x602D58c91c3996c3708cDb966A8cD6F1d87230b1

Taker Actions

Redeem

LP Actions

Subcontract
0xa9a0188a2959

```
0-Fri    1
1-Mon    1
2-Tue    0
3-Wed    0
4-Thu    0
5-Fri    0
```

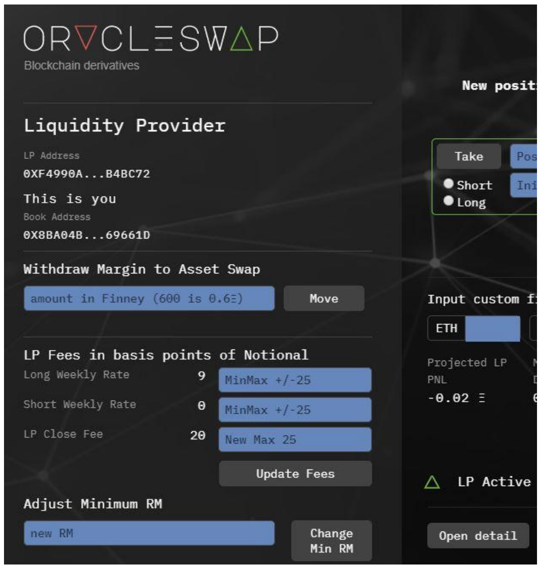
Input custom f

ETH

Your projected P
-0.020 Ξ

Notional
$1445

For the LP, we have the regular options, where the LP can withdraw and add margin


ORVCLΞSWAP
Blockchain derivatives

Liquidity Provider

LP Address
0XF4990A...B4BC72
This is you
Book Address
0X8BA04B...69661D

Withdraw Margin to Asset Swap

amount in Finney (600 is 0.6Ξ) | Move

LP Fees in basis points of Notional

Long Weekly Rate	9	MinMax +/-25
Short Weekly Rate	0	MinMax +/-25
LP Close Fee	20	New Max 25

Update Fees

Adjust Minimum RM

new RM | Change Min RM

New posit

Take | Pos
Short | Ini
Long

Input custom fi

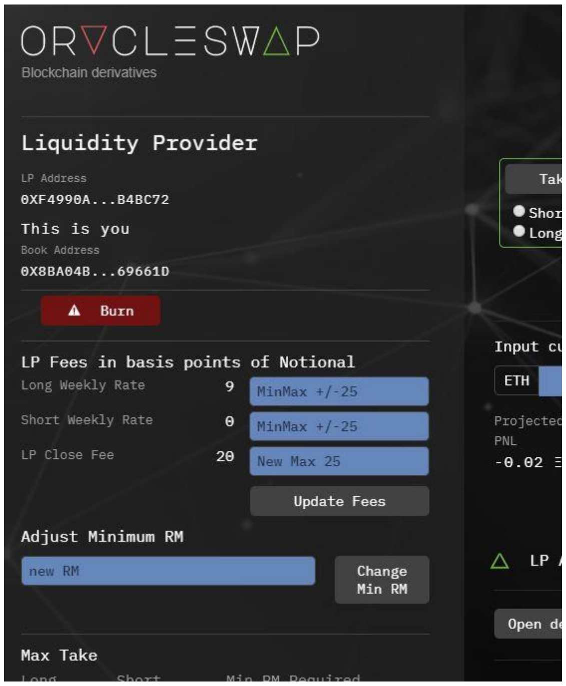
ETH

Projected LP PNL
-0.02 Ξ

LP Active

Open detail

During the first 24 hours after the oracle settlement the LP can only burn, and not withdraw or settle.



ORVCLΞSWAP
Blockchain derivatives

Liquidity Provider

LP Address
0XF4990A...B4BC72

This is you

Book Address
0X8BA04B...69661D

LP Fees in basis points of Notional

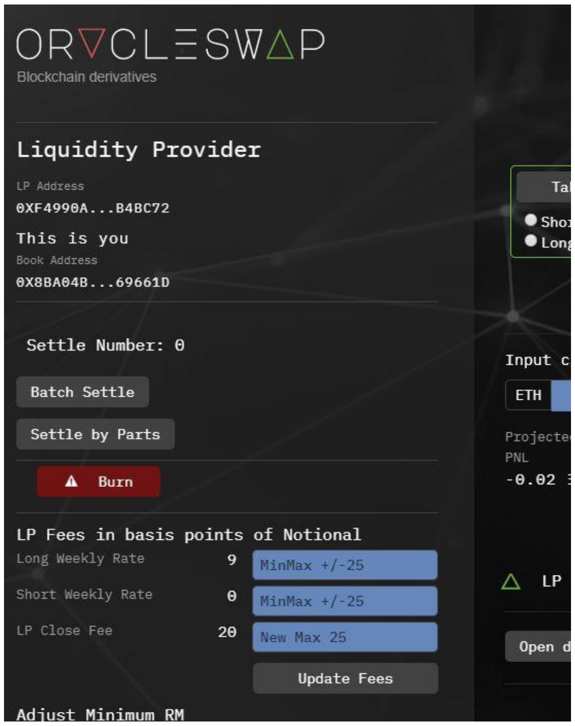
Long Weekly Rate	9	MinMax +/-25
Short Weekly Rate	0	MinMax +/-25
LP Close Fee	20	New Max 25

Adjust Minimum RM

Max Take

Long Short Min RM Required

24 hours after the oracle settlement price update, the LP can settle or burn, but not withdraw or cancel.



ORVCLΞSWAP
Blockchain derivatives

Liquidity Provider

LP Address
0XF4990A...B4BC72

This is you

Book Address
0X8BA04B...69661D

Settle Number: 0

LP Fees in basis points of Notional

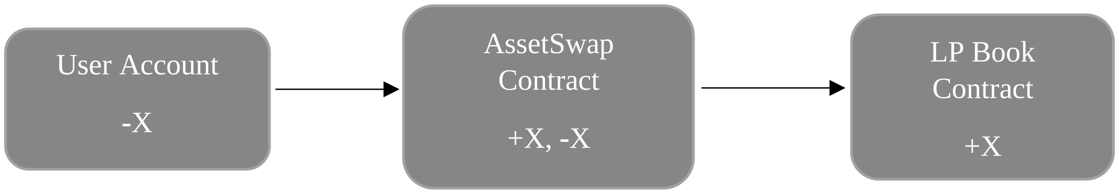
Long Weekly Rate	9	MinMax +/-25
Short Weekly Rate	0	MinMax +/-25
LP Close Fee	20	New Max 25

Adjust Minimum RM

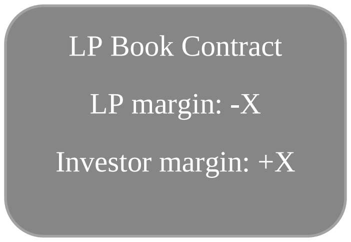
7.18 ETH Payment Flows

Fund

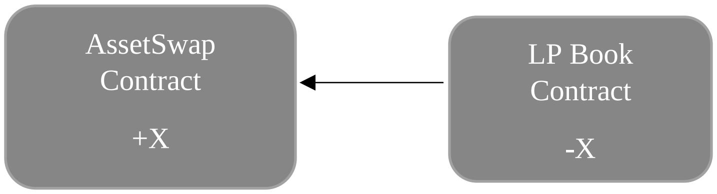
Money sent to the AssetSwap contract goes directly to the LP book contact, and so the ETH resides in the book contract, not the AssetSwap contract. No function accepts ETH from users that stays in the AssetSwap contract.



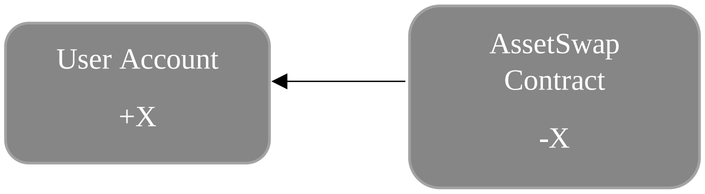
Settlement
At settlement, no ETH is transferred. Player margins are adjusted based on their PNL, where the LP's credit/debit is the inverse of the investor's credit/debit.



Withdraw from Margin
Money withdrawn from margin is sent to the AssetSwap contract and credited to the user's assetSwapBalance. Users cannot withdraw from inactive subcontracts, only redeem, which transfers the entire margin. The ETH then resides in the AssetSwap contract, credited to the user's AssetSwap balance.



Withdraw from AssetSwap
AssetSwap contract balances can only be sent to the message sender's ETH account. It has no 'amount,' rather it transfers all of the ETH credited to the message sender's assetSwapBalance.



8 Security Considerations

This suite of contracts emphasizes simplicity to minimize attack surfaces. While unknown unknowns exist, the following known attacks are addressed.

8.1 General Attack Issues

Re-Entrancy Attacks

The contract has been written with the Checks-Effects-Interactions pattern to avoid reentry attacks. It also uses 'transfer' and not 'send' or 'call' to move ether off the contract. The only function that sends Ether outside the contract is the 'withdrawFromAssetSwap' function which follows the recommended withdrawal pattern. Any attempt to remove Ether from the contracts must go through this function.

Sending and Receiving Ether

As mentioned in the re-entrancy section, the only way ether is sent out of the smart contracts is through the 'withdrawFromAssetSwap' function in the SwapMarket contract. This follows the general security advice of favoring pulling rather than pushing. This function uses Solidity's 'transfer' function, which is considered the safest way to send Ether to an unknown contract, and since 4.x reverts state changes if the transfer fails (we use 5.14). Furthermore, none of the OracleSwap contracts have a fallback function.

Integer Overflow

The contracts utilize one internal procedure to guarantee that the subtraction of uints does not wrap around zero (subzero). Addition, meanwhile, is left alone because in all cases there is no scenario where an item would reach the top of its range. That is, in checking all the potential edge cases, it became apparent that an integer wrap-around is impossible, making the use of a *safemath* procedure redundant. Our subzero function, in contrast, is helpful because in many cases, we want y = max(0, y - x), which this function does as well as making sure a uint does not wrap around its range.

DoS and Gas Limit Attacks

Because there is a finite gas limit that can be spent within a single block, it is important to ensure that no malicious users can manipulate the contract to a state where any transactions will fail by exceeding the gas limit. In the smart contracts associated with OracleSwap, user-exposed functions have either constant execution gas costs or caps. The settle function loops through an LP's set of active subcontracts, but each loop is capped at 200 subcontracts, and the worst-case scenario here is under 5MM gas, well below the 8MM block limit. If there are 545 new subcontracts, the procedure would require 3 sequential settle function calls on the function executing new contracts.

Once a taker is no longer active, they should 'pop' their subcontract from the list of long and short subcontracts, as this is the only way for them to receive back their entire margin. If they fail to do so, however (say if they had zero margin to claim) the LP or anyone else can run this function and remove the subcontract for minimal gas. There is a cap of 2000 total positions, including inactive positions not yet redeemed.

A DoS problem can also occur if a contract requires an external call to finish to progress to a new state, and this external call is corrupted or neglected. OracleSwap does not have an external calls outside the contract.

Timestamp Dependence

It is known that a miner may manipulate the block timestamp up to about 30 seconds. The OracleSwap contracts rely on the block timestamp often to restrict access to certain functions. These functions include when the Oracle may post prices, when new subcontracts can be taken, how often subcontracts may be settled, etc. The scale of all of the time-dependent checks are at a large magnitude (usually hours) and can tolerate manipulation of ±30 seconds.

Front Running

Some Ethereum contracts may have issues where block miners may manipulate the order in which transactions are executed within a block. This a typical problem for market contracts where an attacker might see a transaction for an order and insert their transaction to be executed prior. However, the nature of OracleSwap makes this kind of attack irrelevant. Since forward starting prices are used to open and close a position, pushing someone out of the way to get access to liquidity (say the LP only offered enough to satisfy one new long position), or take an LP's offer when they wanted to retract it, would merely be a minor inconvenience, not costly.

Entropy

No uncertainty or random number generation is used for the OracleSwap suite. This makes the contract inherently resistant to attacks that try to exploit the generation of uncertainty in the smart contracts.

Unexpected Ether

This attack uses the fact that some contracts use this.balance to represent the amount of ether that should be in the contract. As 'selfdestruct' forces a contract to accept ETH, one can put unclaimed ETH into a contract. Further, default and burn payments are effectively left in the contract address, but outside any user's margin or balance, so such a scenario can easily happen. OracleSwap does not reference this.balance in any of its functions.

DelegateCall

In the second Parity Multisig Wallet attack ($150M) uninitialized libraries were accessed via a 'delegatecall' function, allowing the hacker to become the owner of a contract library. The hacker then called the 'kill' function, which froze the contract and all the ETH contained in it. OracleSwap does not use 'delegatecall' and uses no external libraries.

Fallback functions

OracleSwap has a fallback function. The Managed Account contract does, however, in order to withdraw ETH, but this is presented for customization, and seems safe.

External contract referencing non-static contracts.

OracleSwap does not reference external non-static contracts. Initially, the Oracle contract is published without knowing the 3 AssetSwap contracts it is servicing, though it imports the contract code at compile time. The AssetSwap contracts are all published with the oracle's address in its contract constructor, and after that cannot be changed. Once the 3 AssetSwap contracts are published, they are loaded into the

Oracle contract, and cannot be replaced. Users can see these addresses using the method 'AssetSwaps' in the Oracle contract, and also accessing the oracle method in the AssetSwap contract.

Uninitialized Storage pointers

This can cause storage locations for state variables to become transposed, which effectively changes the value of various parameters. The Solidity compiler shows a warning for uninitialized storage variables, and OracleSwap does not contain any.

Short address attack

Some contracts concatenate inputs to save gas, so a short address or parameter then alters a subsequent parameter, such as the amount sent. OracleSwap does not concatenate inputs.

Unchecked call return values

When 'call' or 'send' are used to send ether they return a boolean indicating if the call succeeded or failed, but they do not revert if these functions fail, rather, they simply return false. This can cause the contract to think it sent ether when it did not, which can then allow other contract users to access this unspent ether. OracleSwap uses 'transfer', never 'call' or 'send'.

Constructor Misnaming

If a constructor does not match name of contract it will behave like a function, leading to unexpected consequences. This error involves a constructor not matching the contract name. This is not possible after v0.4.22, and OracleSwap uses Solidity v0.5.14. OracleSwap also has constructor names that match the contract name.

Floating point and precision errors

Solidity does not use fixed-point types, requiring developers to keep track of the precision of the numbers as they are being divided. Functions are designed so that when there is division, the numerator and denominator are in units such that precision is not lost. All prices are input in pennies (e.g., 18100 for $181.00). Other parameters like the closing cost are in units that require dividing by 1e4, etc., to generate the appropriate output.

tx.origin

Contracts that authorize users using the tx.origin variable are susceptible to phishing attacks that trick users into performing authenticated actions on the vulnerable contract. OracleSwap does not use tx.origin.

Public Visibility

The public visibility of functions, where an outsider can see and access a function can allow hackers to manipulate the contract, such as in the first Parity attack ($31M) where a hacker reset the ownership of these contracts and then drained their ether. The only function that changes state that is accessible to anyone is the settlement function. This function is restricted to only be run once after each oracle contract settlement. If someone wishes to execute that for an LP, they are saving her gas. All other functions that involve state changes is restricted by address and must go through the AssetSwap contract. The AssetSwap contract restricts usage to external and is restricted by address.

Other General Code Considerations

If somehow the Oracle or LP disappears, all funds will be made available to users again after 10 days of the admin not updating prices. There is no self-destruct, freeze, or other type of mechanism that would preclude users from accessing their margins.

8.2 Contract Constraints

The only attack surface comes from an evil oracle posting fraudulent prices to inflate the PNL of conspirator positions, presumably the evil oracle's sock-puppet accounts. Everything is vulnerable if the incentives do not align honesty with profit-maximization at every level. OracleSwap's oracle can cheat just as Infura or Bitcoin's miners can cheat, but they are all constrained by their self-interest. Aligning incentives lowers transaction costs, which makes it easier to create contracts that people want to use.

Creating a mechanism that makes honesty the Oracle/admin's dominant strategy involves designing particular payoffs and options, which implies various constraints.

There are several restrictions within the contract code to support the following objectives:

* Make the Oracle settlement price update time predictable so that an evil Oracle cannot sneak in a settlement price while users are not paying attention
* Give users time to react to an Oracle price update so they can burn if the Oracle cheats
* Make it so that users have an incentive to burn a cheating Oracle's payoff rather than let him keep it

To those ends, the following restrictions are in place:

1. Oracle contract updates cannot occur for at least 20 hours after the previous oracle price update
2. Oracle contract settlement updates can only occur when the prior oracle update flagged that the next oracle update will be a settlement update (specifically, the boolean '*nextUpdateSettle*' is marked true). This way a player can know that a settlement price, and thus a settlement, cannot happen for at least 2 days if the next price is not earmarked as a settlement price (it would take 20 hours after the next price to post a settlement price, and then another 24 hours until the LP could settle their book.
3. The book contracts prevent users from settling their books more than once during the settlement period. Specifically, the settlement process cannot commence until the oracle settlement update time is after the last book settlement time (the book update time).
4. Players have at least 24 hours from Oracle settlement update to cure their margins or burn before the LP settles her book.
5. During the settlement period the next Oracle price update must wait at least 48 hours so the LP can settle her book on the oracle's settlement prices. This allows for both the users to cure their margins, burn if the Oracle is a cheater, and also gives the LP time to settle.
6. The oracle contract cannot accept new prices for at least 48 hours, consistent with the back-to-back 24 hour windows for curing and settling.
7. Withdrawals are not permitted during the settlement period. This prevents players from withdrawing their excess margin to zero, which would reduce their default cost if they were left with margin less than ½ RM at settlement (after the PNL attribution). Players with an excess margin greater than 50% of their RM will find burning just as costly as defaulting, but with the

bonus of depriving a cheating Oracle of getting any of their ETH. This increase in the probability of burning conditional upon cheating lowers the incentive for the Oracle to cheat. We want to make burning more probable in the event of a cheat, as this will make the oracle more honest.

8. Cancels are not permitted during the settlement period. The main reason is to prevent the look-back option where one sees Friday prices, and on Saturday finds that the subsequent weekly returns may be bad for them (e.g., they are long and the price fell dramatically Friday evening). This also increases ncreases the incentive to burn if the Oracle posts fraudulent settlement prices.
9. Defaults are paid to the oracle to encourage burning. A player with less than 0.5RM projected in their margin after the PNL attribution, say 0.3RM left over, would find default cheaper than burning, which costs 0.5RM. But the burn fee, and the stolen PNL, would both go to the Oracle, while in default both the PNL and the default fee would go to the oracle. This makes burning that much more painful, increasing the probability a spiteful cheated player would burn.
10. The oracle can change the address of swap contracts. This can be useful if upgrading swap contracts, allowing the oracle to retain its track record of historical price updates. Doing so generates an event log, and this can only be done in a 1 hour window after settlement, so users do not have to worry that an oracle would be doing this frequently, or right before a settlement. An absent oracle would prevent settlement, and after 10 days allow all AssetSwap players to withdraw their entire margins.
11. The oracle/admin can add and remove addresses from the AssetSwap and Oracle contracts. This is useful in changing wallets and seed phrases. It does not change the oracle/admin's opportunities or incentives.

The LP book is accessed by looking at the address of the book contract created by the message sender. Thus one must have the private key to 0x123...456 to access the book contract associated with the LP address 0x123...456. These functions include

- Updating fees
- Adjusting minimum RM for subcontracts
- Burning their book
- Withdrawing margin from the book contract
- Setting the book up for cancellation in 4 weeks

The taker to the subcontract must access the contract via their taker address to execute the following functions

- Withdraw
- Burn
- Inactivate the negligent LP

As all withdrawals are sent to the AssetSwap contract into the message sender's balance. It can then be withdrawn, where the message sender's address points to a balance in the AssetSwap contract, and any money in that address is sent out of the AssetSwap contract to the user's off-contract account.

One can fund their margin using any Ethereum account address.

9 Definitions

AssetSwap Balance: This balance is a way station for withdrawals from margin, used to prevent re-entry attacks. The AssetSwap balance does not count towards a player's margin. A player wishing to take ETH out of their Margin first withdraws from their margin to their AssetSwapBalance, and then from this Withdrawal Balance to their off-contract public address. User money is transferred from book margin to the AssetSwap only via withdrawals and redemptions.

AssetSwap Contract: Each asset serviced by the Oracle Price Contract has a separate AssetSwap Contract. For example, BTC will have a separate contract than the SPX. They are all identical, just with different reference assets and leverage ratios.

Basis: The adjustment in a CFD or swap contract that accounts for interest rates, dividends, storage costs, convenience yield, and risk premium. This is implicit in the difference between a futures/forward and cash price, but for a swap, there is only a cash price, so the basis is applied as an effective funding rate, applying symmetrically to longs and shorts (subtracted from the long return, added to the short return). In this contract, the basis is merely implicit within the differential between the long and short funding rate, which also includes a target rate that is identical for longs and shorts for a specific asset. The basis equilibrates long and short swap demand. Funding rates, which include the basis, are constrained to be between -1 and +1% per week for crypto, 0.25% for SPX. Once a subcontract is taken, its funding rate is fixed for its duration.

Book Contract: When a player first posts as an LP, the asset contract creates a unique contract for that investor's contract address that will hold all that LP's counterparties. While all players always interact with the base Asset Swap Contract, this then transacts with the active LP Book Contracts. All margin ETH is held within the Book contract, while ETH in AssetSwap Balances are held in the Asset Swap contract.

Burn: A counterparty can burn their contract, which then prevents their counterparty from receiving their debit at the subsequent Weekly Settlement. The payable burn fee of RM/2 is not attributed to any player margin, so it is inaccessible and effectively burned. A burn costs as much as a default, though the payment mechanism is different, in that the burn requires the player to send RM/2 but then does not debit the player's margin at redemption; a default debits the players margin at redemption (or, for the LP, at the final settlement).

Business Day: A business day corresponds to a New York Stock Exchange business day and are the only days that an initial subcontract price is set, or that a Weekly Settlement can occur. The Oracle Price Contract does not record non-business day prices, though players can take on these non-business days and their subcontract will initialize at the next business day. Business days thus exclude weekends and about nine holidays. These can be found by Googling 'NYSE holidays' which are published three years in advance. Half-day holidays will use the 4 PM crypto price, though the SPX price will use the 1 PM official closing price.

Cancel: A counterparty who wishes to terminate the contract initiates a cancel. This must be done before the Friday Oracle Price Update around 4:15 PM. Intraweek cancels are charged the maximum LP closing fee of 1.0% for crypto, 0.25% SPX. LPs cannot cancel intraweek.

CFD: A Contract-For-Difference is like a futures contract in that counterparties put up a fraction of the notional, their margin, and use an asset price for generating a mark-to-market PNL on that notional amount. Unlike futures, the reference is not a separately traded price, but rather the cash price, so to account for the basis we see in the futures market, long and short positions are charged different funding rates in CFDs. These are also called total return swaps, perpetual swaps, or just swaps.

Closing Price: Crypto prices are taken around 4 PM ET on NYSE Business Days from various but unnamed Bitwise-approved exchanges. We use the official close for the SPX, which refers to 4:00 PM but is finalized around 4:10 PM.

Default: If a counterparty's margin is below their subcontract's Required Margin at settlement, the subcontract defaults and is terminated. This can only occur at the Weekly Settlement, as investor withdrawals below a subcontract's RM are not allowed and a PNL debit, can only be applied at the weekly settlement. Players should be able to anticipate if they have enough ETH to cure their margin prior to the Oracle contract settlement day update at 4 PM, and so by canceling beforehand avoid this charge.

Excess Margin: Total or Actual Margin minus the Required Margin. This amount can be withdrawn at any time outside of the settlement period.

Fed Funds rate: The overnight interest rate at which US banks lend to each other. It serves as the basis for the cost of funds among financial institutions and is the opportunity cost of a US dollar.

Forward-Starting Price: a contract price set after an agreement to open or close a position is made, such as the next closing price (market-on-close) or the next day's value-weighted average price. In OracleSwap, a take looks to the Oracle to determine the current price slot (e.g., 2 for Tuesday, the third slot), and then sets the initial or ending price day as the next price day. If a new contract, and the next day is a settlement day (e.g., Friday), it will not process at that week's settlement, but use the settlement price as the initial price in the subsequent week's settlement.

Funding Rate: The weekly rate applied to the Taker. This rate is subtracted from the return for the taker and applied to the notional amount. These can be negative, in which case, would add to the Taker return. LP's set their own funding rates, and by adjusting the difference between the long and short funding rates, equilibrate their long and short balances.

Gwei: 1e-3 Szabo, or 1e-9 ETH. 1e-3 Szabo, or 1e-9 ETH. Pronounced 'gway.' Anyone developing their own version of OracleSwap should use ETH as the unit of denomination, and Finney as the subunits. Here, Gwei are equivalent the subunits of Szabo.

Leverage Ratio: The OracleSwap's leverage ratio is the ratio of the notional to the RM for that asset. It can be thought of as the inverse of the required margin ratio. It is set such that the RM should cover 99.5% of weekly events, and is currently 2.5 for the ETHUSD and BTCUSD, and 10.0 for the SPX. This corresponds to a margin ratio of 40% for crypto, 10% for SPX.

LP: A liquidity provider. They post an amount available for long or short takers. They are paid via funding rates on their gross exposure for the unpredictable risk generated by ending up net short or long.

Market-on-Close: A Market-On-Close (MOC) order is a non-limit market order that is executed at some official closing price, which is at 4 PM for US stock markets. MOC orders do not specify a price, as these are forward-starting and thus unknown at the time of order. OracleSwap uses quasi-MOC orders, in that the oracle samples prices from 4:00 - 4:03 PM.

Margin: This is the total ETH attributed to a player. A Taker with several different subcontracts will have a separate margin for each, while an LP will have one margin for all her various counterparties. This amount is in the LP book contract. It takes only one function to fund into margin, but two functions to withdraw (first from the margin, secondly from the AssetSwap balance).

Max Long/Short Take: this is the most a taker can take, long or short, for a given LP. It is a function of the LP's current book. For example, if the LP shows 100 Margin with no positions, a Taker can only go long or short 50. If the LP shows a margin of 100, with the LP currently long 100, Takers can only go short.

Max Short Take = Max{0,Min(LP excess marg, ½ Total LP Margin + LPShort – LPLong)}
Max Long Take =Max{0,Min(LP excess marg, ½ Total LP Margin + LPLong – LPShort)}

Net Margin: An LP's Required Margin is netted, meaning it is the difference between the gross long and short for an LP. For example, if the gross long is 80, gross short is 50, the net margin is 30. The LP's RM is equal to their net margin, abs(long – short). Only the LP has an RM based on a Net Margin. For a taker, there is no netting if they take offsetting positions within an LP's book.

Notional: Notional is the amount applied to the reference asset return to generate the ETH PNL. It is generated by multiplying the RM by the Asset Contract's specific Leverage Ratio and then the ETH price in USD. This is in USD per ETH.

Oracle Price Contract: The Oracle Price Contract warehouses the asset prices of all AssetSwap Contracts serviced by OracleSwap. It contains all the information needed to calculate subcontract returns, and so holds the closing prices for the current week (i.e., last Friday to the current date). The Oracle posts prices between 4 and 5 PM New York City time. At settlement update, the oracle contract pushes the asset returns used for generating PNL to each AssetSwap contract.

Player: Synonym for an OracleSwap counterparty, investor/taker or LP.

PNL: Profit-n-loss, the cash flow between two parties of a subcontract, where the loser/debtor has a negative PNL, and the winner/creditor has a symmetric positive PNL.

Redeem: This function moves the investor's ETH to the AssetSwap contract, from where it can be withdrawn to the investor's account. It is the only way to withdraw funds after a subcontracts last settlement, or if the LP's book is inactive or in default.

RM: Required Margin. Each subcontract has a specific RM for its life, in units of ETH. RM must be greater than or equal to the LP's minimum RM. This sets the notional amount via the leverage ratio, so given the RM we have a notional and vice versa.

Settlement: Settlement is applied each weekend by all LPs with active subcontracts via the Settlement Function. This uses the Friday closing prices recorded in the Oracle Price contract to the subcontracts and then transfers the resulting PNL from the debtor to the creditor. If Friday is an exchange holiday (e.g., Christmas), the prior business day, Thursday, will be the settlement day.

Settlement Period: The time between when the Oracle contract settlement prices are updated Friday 4 PM and when the LP runs the settlement function 24-48 hours later. Withdraws and cancels are not allowed so that users who see a fraudulent price report by an evil Oracle are then properly incented to burn rather than subject themselves to another week where they will probably be cheated again. Users can fund margin in this period. Users need to cancel before the settlement period, before the Oracle contract update on settlement day, to exit at the next settlement.

SPX: The SPX is the index for the S&P500 portfolio, a value-weighted index of the top 500 US companies. It is a cash index that excludes dividends. In contrast, the SPY is an ETF that trades on stock exchanges, and so its price is dividend adjusted. This affects the basis in the futures market, so the basis is Fed Funds minus the dividend rate for the popular SPX futures traded on the CME.

Szabo: 1e-6 ETH. This web version uses Szabo because it is not meant to be economically meaningful, just a way to see how it works. Anyone developing their own version of OracleSwap should use ETH as the unit of denomination.

Price Day: The day of the first price for a subcontract. It is determined by the Oracle contract, which knows what the next day of the week will be. Thus, if it is Tuesday prior to the oracle price update in the 4-5 PM time period, the start day will by Tuesday; if it is Tuesday after the oracle price update, the start day will be Wednesday. For a position with an intraweek close, the price day is actually the ending day price, and so the return from start day to settle is subtracted from their returns.

Subcontract: a bilateral agreement between a Taker and LP. It has a specific RM for its life, as well as a specific funding rate and closing fee determined at trade instantiation. An LP will have many subcontracts. Each taker/investor will have the LP as his counterparty.

Subcontract ID: A subcontract is uniquely identified in in the AssetSwap contract via an LP's address and subcontract ID (which is in bytes32). The subcontract uses the LP address to find the specific book contract, then within that contract, the subcontract ID to find the particular subcontract. The Subcontract ID is a hash function based on the block time and size of the LP's book at that time, ensuring each subcontract has a unique ID.

Swap: A swap in the context of this document is a CFD and has no expiration (e.g., a perpetual swap). A swap account nets a user's positions to generate their required margin and total PNL, and its mark-to-market is based on the cash price of its reference assets; it also charges a funding rate to long and short positions. Prime brokers use these to provide hedge funds a way to avoid stamp taxes and allow them to mask their positions for strategic reasons. For equities, their CFDs are often in swap accounts—swapping the cashflows from long and short equity positions—and 'swap' vs. 'CFD' terminology is a matter of preference.

Taker: A counterparty to a subcontract that takes an LP offer. Also known as the investor. The taker determines the side (long or short) and size (RM).

Termination: When a subcontract ends for any reason—burn, cancel, default—its termination occurs at the Friday settlement. This then sets the subcontract's RM to zero, allowing the Taker to withdraw his total margin, while the effect on an LP's margin is ambiguous. A book that is terminated has the variable bookStatus set to 3, allowing all takers to redeem, and the LP to withdraw, regardless of their required margin.