Super Tranche

hpark2017

October 2022

1 Glossary

- `tVault`: The asset to be splitted is wrapped into an associated ERC4626 `tVault`. For example, an stETH asset can be wrapped by $\texttt{tVault}_{stETH}$, to be splitted by $\texttt{senior}_{stETH}$ and $\texttt{junior}_{stETH}$.
- asset: The volatile asset to be splitted, and wrapped by `tVault`.
- underlying: The numeraire for which the value of asset is denominated in.
- R_f: the *promised* fixed returns for senior token holders, per unit timestep.
- R_{r_i}: The *real* returns of the asset denominated in the underlying of the `tVault` at timestep i. For yield generating instruments like $\texttt{tVault}_{stETH}$, the returns would be denominated in ETH, and for spot assets like $\texttt{tVault}_{ETH}$, it could be denominated in USD.
- P_{js}: Price of `junior` denominated in `senior`
- P_{su}/P_{ju}: price of `senior/junior` denominated in underlying.
- P_{sv}/P_{jv}: price of `senior/junior` denominated in `tVault`.
- P_{uv} Price of underlying / `tVault`. It's change will dictate the returns of a unit amount of `tVault`.
- value P_{js},P_{su},P_{ju}: prices as determined by the pricing rule
- mark P_{js}: effective prices as determined by the AMM.

2 Introduction

We introduce a mechanism for perpetual, permissionless(PP) tranches. pp-Tranche is a general volatilty segregation module, where it can split any volatile asset into two assets(senior and junior) with differing volatility profiles. The asset to be split could be a yield generating token i.e (stETH, where the denominated underlying is ETH), or any assets that realizes volatility(i.e ETH) that ideally has high market cap and where the asset/underlying(numeraire such as USD, equities, bonds, commodities, etc) pair has an oracle onchain. Its permissionless nature allows anyone to instantly create a market for yields or capital-efficient synthetic derivatives for any (combination of) instruments with an exchange rate oracle, while its perpetuity mitigates liquidity fragmentation and provides an efficient user interface. Our proposed design serves to satisfy the need of the following agents.

For splitting yield generating tokens:

- Agents who wish to directionally speculate on yields
- Agents who wish protected yield or for some yield source(from variable to fixed)
- Agents who want leveraged yield to a yield source
- Yield instrument(s) holders who wishes to amplify yields on top of their instruments via fees generated from liquidity provision.

An example instrument of use could be any yield bearing vaults that has a exhibits variability in its yield, such an L1 asset staking derivative, decentralized options vaults (DOV), or lending pool supply positions.

For splitting general assets:

- Agents who want leveraged exposure on the asset.
- Agents who want dampened exposure on the asset, while earning yields.
- Agents who want exposures to synthetic assets
- Agents who want to provide single-asset liquidity.

An example of such instrument is ETH, BTC(any volatile asset that can access accurate price oracles), where the `underlying` to be synthesized could be off-chain assets such as USD, bonds, commodities or equities.

2.1 Simple Numerical example

A `tVault`, which can be composed of a singleton or of a pool of `asset`, and where its total value denominated in `underlying` is always quantifiable by an exchange rate oracle, can be separated by a `Splitter` into a `senior` and a `junior` token, by a fixed ratio $0 \leq w \leq 1$, such that the invariant(in quantity) $(1-w)\texttt{senior} + w\texttt{junior} = 1\texttt{tVault}$ always holds. If w is 0.3, for example, 1 `tVault` will always split to 0.7`senior` and 0.3`junior`. Equivalently, the `Splitter` will only accept $(1-w)\texttt{senior} + w\texttt{junior}$ when redeeming for 1`tVault`. w would be a definable parameter for each `tVault`, and would characterize the degree of leverage `junior` incurs.

This pair will trade in an AMM under mark P_{js}, such that a trader who wants $\texttt{senior}_{asset}$ of $\texttt{tVault}_{asset}$ would have to go through the following procedure

1. mint $\texttt{tVault}_{asset}$ with asset
2. split by the ratio w_{asset}
3. swap the remaining `junior(senior)` to `senior(junior)` in the AMM

Equivalently, a trader who wants to trade back to $\texttt{tVault}_{asset}$ from $\texttt{senior}_{asset}$ could do so by the opposite, he would trade $\texttt{senior}_{asset}$ back to the appropriate pair ratio, merge, and redeem for $\texttt{tVault}_{asset}$

3 Pricing

It is then natural to ask how should the pair be priced. We apply a programmatic pricing model for both $P_{ju}$and P_{su}, where then the value $P_{js} = \frac{P_{ju}}{P_{su}}$ can be then be derived.

To price the junior/senior pair, it is useful to first reason on the constituents that determine the true value of each `junior` and `senior` tokens, denominated in `underlying` of the `tVault`. The pricing model should a) ensure that the return profiles of both `junior` and `senior` lie in the construct of which they are defined, i.e that seniors have the primary claims to returns while juniors absorb volatility and b) ensure that, when every supply of `junior` and `senior` are redeemed for `underlying` under this computed value, the system remains solvent

In light of these considerations, we present the following pricing mechanism. For a given $\texttt{tVault}_{asset}$, a $\texttt{senior}_{asset}$ would have a fixed return profile R_f(determined at tranche inception and compounded per unit time), such that it's value at time t, as denominated by the $\texttt{underlying}_i$ of $\texttt{tVault}_{asset}$ would be defined by the following formula:

$$priceofsenior/underlying := P^t_{su} = I(R_f)^t$$

while its pair $\texttt{junior}_{asset}$'s would be defined as the pro rata share of the remaining assets after all circulating $\texttt{senior}_{asset}$ at t, S_{s_t}, has redeemed for P^t_{su}.

$$priceofjunior/underlying := P^t_{ju} = \frac{A_t - (I(R_f)^t S_{s_t})}{S_{j_t}}$$

and at t = 0(when tranche is initialized),

$$P^0_{su} = P^0_{ju}$$

$A_t := I \prod_{i=1}^{t} R_{ri}$ denotes the total assets of $\texttt{tVault}_{asset}$ at time t denominated in `underlying` of $\texttt{tVault}_{asset}$. (R_f) is, for every incremental t, the fixed return rate promised to seniors, and R_{ri} is the *real* return at timestep i. I is the inception price of both $\texttt{junior}_{asset}$ and $\texttt{senior}_{asset}$ when $\texttt{tVault}_{asset}$ is created. S_{s_t} and S_{j_t} respectively denotes the circulating supply of $\texttt{senior}_i$ and $\texttt{junior}_{asset}$ at time t. (Without loss of generality we subsequently set $I = 1$).

Using the previously defined invariant property we have that $S_j = \frac{w}{1-w}S_s$, and making no other assumption other than the fact that the total underlying asset A_t held by `tVault` at a given point is simply $(S_j + S_s)P_{vu_t}$, where P_{vu_t} is the exchange rate of `tVault` and `underlying` at time t, P_{ju}^t can also be represented as

$$P_{ju}^t = \frac{w}{1-w}(\frac{\prod_{i=1}^t R_{ri}}{w} - (R_f)^t)$$

This representation of P_{ju}^t eliminates its dependency of either S_j and S_s, allowing the pricing to be agnostic to supply, and is solely a function of the returns accumulated until timestep t. This property ensures the system, under this exact pricing, can withhold arbitrary amounts of influx and outflux of capital while remaining solvent, which implies that all minters and redeemers of $\texttt{junior}_i$ and $\texttt{senior}_i$ tokens would be guaranteed their pro rata share regardless of their entry point t (thus its perpetual nature). However, **this assumes that the system always mints and redeems $\frac{w}{1-w}$ junior for every 1 senior minted**. So it is necessary for a user who wants a senior(junior) token to first mint a `tVault`, split it by the predetermined ratio w, and find a suitable counterparty to swap her remaining junior(senior) token for an appropriate amount of senior(junior) token. Then the pricing rule P_{ju}^t and P_{su}^t finds their utility by determining the *price of junior denominated in senior*

$$P_{js}^t := \frac{P_{ju}^t}{P_{su}^t}$$

$$= \frac{\prod_{i=1}^t R_{ri} - w(R_f)^t}{(1-w)(R_f)^t}$$

which can compute the amount of senior(junior) will obtain by selling her junior(senior).

It is important to understand the behavior of the pricing rule given its parameters. By rearranging equations we can answer the following questions: how does the price of junior/senior change as the real returns changes? How does $P_{js}^{t+1} - P_{js}^t$ differ as R_{r_t} changes? How dependent is it on the previous timestamps? Noticing that P_{js}^{t+1} is simply $R_{r_{t+1}}\frac{\prod_{i=1}^t R_{ri} - w(R_f)^{n+1}}{(1-w)R_f^{n+1}}$ we can be explicit in the relationship of the real returns at timestep $t+1$ and the difference in prices at $t+1, t$

$$P_{js}^{t+1} - P_{js}^t = \frac{1}{(1-w)R_f}\frac{\prod_{i=1}^t R_{ri}}{R_f^t}(R_{r_{t+1}} - R_f)$$

. We can see that the new price at t+1 is dependent on the difference between the real returns and promised returns at time $t+1$, weighted by the cumulative difference up until time t. This implies that an economic rational trader who believes $R_{r_i} \geq R_f$ for $t \leq t+n$ for some time period n would trade to obtain junior for senior tokens, and vice versa.

This deriviation merits some attention, as it(put informally) gives rise to an interesting phenomenon where for $i \in 0, 1, 2, 3...$ if R_f is set such that

$\mathbf{E}[R_{r_i}] = R_f$, the `junior/senior` pair will essentially trade like a (loosely pegged) stablecoin, where the LPs suffer no adverse selection. (Show price comes back, give example)

We can also easily verify that under this pricing rule, all traders who enter at any arbitrary time can realize their profit by showing that the unit sum of profits for seniors and juniors over n timesteps, normalized by the ratio coefficient w, equate to the sum of a single `tVault` token over n timesteps. For a single `senior` and `junior` token, their profit from timestep t to $t+n$ can be expressed as

$$Pf_s := (R_f)^{t+n} - (R_f)^t$$

$$Pf_j := \frac{w}{1-w}\left(\frac{\prod_{i=1}^{t+n} R_{ri}}{w} - (R_f)^{t+n} - \frac{\prod_{i=1}^{t} R_{ri}}{w} + (R_f)^t\right)$$

Since $\frac{1}{w}$`tVault` always splits to 1 senior and $\frac{1-w}{w}$ junior,

$$Pf_s + \frac{1-w}{w}Pf_j = \frac{\prod_{i=1}^{t+n} R_{ri} - \prod_{i=1}^{t} R_{ri}}{w}$$

which is the profit for $\frac{1}{w}$`tVault` over n timesteps starting from timestep t.

3.1 Additional Algebra

We also show that P_{sv} and P_{jv} can be derived from w and P_{js}, for which values are valuable for computing profits of tranches denominated in vaults in subsequent sections. Given a `tVault` quantity $a + a\frac{1-w}{w}$, where a and $a\frac{1-w}{w}$ respectively represents the quantities of `senior` and `junior` we know that this amount of `tVault` can be split as the following

$$aP_{sv} + \frac{1-w}{w}aP_{jv} = a + a\frac{1-w}{w}$$

substituting $P_{jv} = P_{js}P_{sv}$, we arrive at the following solution for P_{sv}

$$P_{sv} = \frac{1 + \frac{1-w}{w}}{1 + P_{js}\frac{1-w}{w}}$$

and equivalently for P_{jv},

$$P_{jv} = \frac{1 + \frac{1-w}{w}}{1 + \frac{\frac{1-w}{w}}{P_{js}}}$$

4 Swapping

Recall that an agent who wishes to gain further yield on his asset can split it into a senior and junior by ratio w and provide liquidity into the associated AMM. Constructing a pool for a senior/junior pair rather than a separate pool for senior/underlying and junior/underlying is useful for two reasons. a) the LPs need not provide a separate pair and provide single asset liquidity b) under the conditions where the expected(average) real returns over time roughly equals the fixed return, the senior/junior essentially trades like a loosely pegged stablecoin pair, where LP profits would be predictable.

Using the liquidity provided by the LPs, a user who wishes to convert his asset(i.e ETH or stETH) to a senior(junior) token can do so by

1. minting `tVault`$_{asset}$ with asset
2. split as accordance by the ratio w
3. swap the remaining `junior`(`senior`) to `senior`(`junior`) in the AMM

A user who wishes to convert his `senior`(`junior`) token back to his asset can do so by

1. swap back in the AMM until the ratio is met
2. unsplit(merge) to a `tVault`$_{asset}$
3. redeem `tVault`$_{asset}$

It can be seen that the illustrated steps, when there is infinite liquidity in the AMM, can be respectively abstracted as a minting and redeeming module for `senior`(`junior`) tokens where the AMM serves as the key element, and it necessitates that the AMM prices the junior/senior pair as defined by the pricing rule.

The design space for this AMM is vast, each manifesting a unique set of trade-offs. We outline two specific instances that have been implemented, an oracle based and an reserved based AMM.

4.1 Oracle Based AMM

This is a rather simple trading module, where value P_{js} determined by an oracle is the sole input when computing quantity to be pushed for a given quantity to be pulled.

For an LP, the pool will structurally resemble a vault, where an exchange rate determined by the ratio of liquidity share supply and liquidity pool value determines how much pool value is withdrawed/deposited for an unit share. Liquidity pool value is represented as the following sum.

$$P_{sv} * SeniorSupplyInReserve + P_{jv} * JuniorSupplyInReserve$$

This value will be determined by the profit or loss the "always taking the opposite side" trading strategy makes in addition to the fees accrued from takers' trades.

4.1.1 Advantages of an Oracle Based AMM

Advantages of this AMM model lies in its simplicity and its displacement of arbitraguers. When oracles are reliable and accurate, this implies that there would be less informed takers to extract value from the pool for assets when traders normally don't have an edge in predicting P_{js}. Therefore this model is more suited for general volatile asset/underlying pairs where reliable oracle are available along with well documented frontrunning mitigation techniques(i.e asset: ETH underlying USD).

4.2 Path Independent AMM

This is a reserve based AMM(our implementation exhibits non uniform liquidity as in univ3). Such a model would be preferred over an oracle-based AMM when a) the oracle that computes value P_{js} is not reliable b) when P_{js} is highly predictable (If P_{js} is highly mean reverting, a trivial "always buy low sell high strategy" can drain liquidity from the pool. A path independent, slippage sensitive AMM would ensure that the LP's inventory negating fees be a deterministic function of price).

The slippage a trader incurs while swapping in this AMM determines the discrepancy between the mark P_{js} and the value P_{js} for the trader, which is not ideal for the predictability of the profitability of tranche tokens. Our goal is then to design a system that *incentivizes* value P_{js} to (almost) equal mark P_{js} without enforcing strict pricing in the AMM. (we henceforth term this phenomenon the *internal peg*) We propose two mechanisms that mitigate deviations from this internal peg; a) Indirect Aribtrage and b) Bounded trade enforcement.

For markets where there is a clearly defined true value(true in the sense that, given the context, the system can mint and redeem with prices in accordance to the given value) of the asset, there generally exists a party of arbitrageurs that are economically incentivized to set the mark price approximately equal to the valued price. However, the presented system doesn't exhibit a clear path for arbitrage, since it solely accepts a predetermined ratio of senior/junior to split or merge back into a `tVault`. For example, if an arbitrageur notices that the mark P_{js} of junior/senior in the AMM is **higher** than the value P_{js} computed by the pricing rule, he *should* be able to go through the following procedure

1. mint $\texttt{tVault}_{asset}$ with asset (Numerical example: mint 100 $\texttt{tVault}_{asset}$ with 100 asset)

2. split as accordance by the ratio w (Numerical example: Split 100 $\texttt{tVault}_{asset}$ to 70 $\texttt{senior}_{asset}$ and 30 $\texttt{junior}_{asset}$)

3. swap the remaining `junior` to `senior`in the AMM (Numerical example: swap 30 $\texttt{junior}_{asset}$ to $30 * 1.2 = 36$ $\texttt{senior}_{asset}$, where 1.2 is the mark P_{js}, for a given value P_{js} of 1.1)

4. redeem the `senior` for more $\texttt{tVault}_{asset}$ at the valued price $P_{sv} := P_{su} * P_{uv}$. Since the amount of $\texttt{tVault}_{asset}$ would be greater then that in step 1, this will result a profit. (Numerical example: Now the trader has $70 + 36 = 106\texttt{senior}_{asset}$. For the given $w = 0.7$ and value $P_{js} = 1.1$, $P_{sv} = 0.97$, where he will then redeem for $106 * 0.97 = 102.9$ $\texttt{tVault}_{asset}$, where he will collect 2.9 $\texttt{tVault}_{asset}$ as profit.)

However, step 4 is not allowed as the system's ratio invariant only allows the `senior` to be accompanied by a $\frac{1-w}{w}$ amount of junior tokens during the redemption process. We thus present an *indirect* arbitrage mechanism that allows the system to expel excess demand(supply) for an overpriced(underpriced) tranche.

4.2.1 Indirect Aribtrage

We should first acknowledge that there is always going to exist some degree of market force that will set value P_{js} close to mark P_{js}. For example, if a senior token holder who expects R_f monthly returns sees that given the current inefficiency she can realize the profit(by selling her senior to an appropriate ratio of senior/junior and merging+redeeming) that are meant to be made at a much forward time in the future, it is economically rational for her to do so.

However, this does not prevent the circumstance where the trader would "hoard" her tranche expecting the inefficiency to subsist. Hence the system needs a stronger market force that can restore this internal peg. This generally requires participants who can introduce exogenous capital to go through the steps 1-4 outlined above.

We instead propose to replace step 4 with a slight modification. Instead of redeeming `senior` for $\texttt{tVault}_{asset}$, the system will allow it to redeem it for $\texttt{dVault}_{asset}$(d stands for debt). In the numerical examples above, the arbitrageur will instead receive $102.9\texttt{dVault}_{asset}$. These $\texttt{dVault}_{asset}$ represents a promise to pay the arbitrageurs(by converting it to $\texttt{tVault}_{asset}$ 1:1) the yet to be realized profit, which occurs when either of the two circumstances are met;

- A `junior` holder wishes to redeem their `junior`s for $\texttt{tVault}_{asset}$. Using the `senior` liquidity used to mint $\texttt{dVault}_{asset}$ she can do so without needing to swapping part of her `junior` back to `senior`. Instead, she can pair her `junior` with the `senior` that has minted $\texttt{dVault}_{asset}$ in accordance to the the ratio w and redeem $\texttt{tVault}_{asset}$ without incuring any slippage. For every `senior` the arbitraguer has redeemed for $\texttt{dVault}_{asset}$, the `junior` holder needs to redeem $\frac{1-w}{w}$ `junior` for all the $\texttt{dVault}_{asset}$ to be convertible to $\texttt{tVault}_{asset}$.

- A trader who wishes to purchase `senior` with $\texttt{tVault}_{asset}$ directly without needing to split/swap in the AMM, can do so by consuming the `senior` liquidity that was provided by the arbitraguer. She gives the system $\texttt{tVault}_{asset}$ in return for `senior`, and this swap will occur at a the value P_{sv} that the arbitraguer had redeemed at.

These `dVault`$_{asset}$ are always collateralized as, during circumstances when the two events fail to occur, the arbitrageur can always *unredeem* his `dVault`$_{asset}$ back to `senior`.

4.2.2 Bounded trade enforcement

A relatively simple mechanism, this enforcement will revert the following category of trades. Define some parameter $\delta \geq 1$ and a price range value $P_{js} * \delta$, value P_{js}/δ,

- Pre-trade mark P_{js} is within the range and the post trade mark P_{js} lies outside the range.
- Pre-trade mark P_{js} is outside the range and post-trade mark P_{js} is farther away from the range.

It is trivial to see that mark P_{js} quotes that relies solely from oracles without liquidity sensitivity and bounded trade enforcement will result in adverse selection from malicious arbitrageurs when the oracle feed incorrect or delayed, which can often be the case for long tail assets. The bounded trade enforcement effectively upperbounds this loss by some amount proportional to δ.

4.2.3 LPs profitability

In this reserve based AMM, LPs would be highly incentivized to supply in a market where the expected real returns of the `tVault` per unit time is roughly equal to the promised return per unit time $\mathbf{E}[R_{r_i}] = R_f$, as such markets would allow prices of pairs `junior/senior` to almost always be mean reverting(as an example, consider a variable yield bearing instrument that is decomposed to a fixed(senior) and a variable(junior) token where R_f is set such that it roughly equals the variable rates on average)

5 Oracles

Oracle manipulations and frontrunning are well-documented attacks for systems that modify states based on off chain data. While the oracle based AMM are subject to these perils, the path independent AMM circumvents these attacks as the swaps are conducted in an liquidity-sensitive AMM with non uniform(limit orders/concentrated) liquidity where the pricing is not enforced but rather incentivized via indirect arbitrage and outside bound reverts. However, as the incentivization layer relies on external data to compute value P_{js}, oracles are still required.

5.1 Yield Instruments

Yield instruments implemented under the ERC4626 standard that are to be wrapped in `tVault` allow generic composability(and hence the permissionless creation) via the `totalAsset` default oracle and `afterDeposit` and `beforeWithdrawl` hooks. Indeed, the implementation of the `totalAsset` function for a given token might not be suitable for a standalone oracle, and thus would serve to be the primary bottleneck to safety and the trade-off for permissionlessness. While our implementation on default uses the time-weighted average `totalAsset` for state updates, a distinction between *verified* and *unverified* `tVault` is to be made on whether the `totalAsset` under the TWA setting is to be trusted.

5.2 General Volatile Assets

We utilize oracles that are well documented and tested such as Chainlink price oracles.

6 Lending Pool

Recall that the system also allows traders to directionally speculate on yields. This could be achieved via the following procedure(implemented atomically via flash minting `junior` or `senior`)

1. mint $\texttt{tVault}_{asset}$ with asset
2. split by ratio w
3. swap the remaining `junior` to `senior` in the AMM
4. use `senior` as collateral to *borrow* more `junior` tokens, and repeat step 3.

In this example, if P_{js} decreases, the trader will earn more seniors when he deleverages his position. And as shown in section 3, P_{js} decreases when the the promised return R_f is greater then the real return R_r for subsequent timesteps.

This mechanism necessitates a lending pool for each tranche. Traders can supply their tranche tokens in this lending pool for extra yield.

7 Use cases

7.1 Yield speculation and risk compartmentalization

Let us illustrate the system flow when $tVault$ wraps a yield generating asset such as $stETH$. Alice holds 100 stETH where each stETH is currently worth 1ETH. As per the outlined needs, she has several options.

- If Alice wants to directionally speculate on yields, (in particular if she wants to bet on yields decreasing) she can deposit 100 stETH to mint x senior(stETH) and 100 - x junior(stETH), swap junior to senior, and with the output senior as collateral borrow junior and repeat the swap procedure until she is satisfied with her leverage.

- If Alice wants to earn a fixed yield from the current variable stETH yield, she can deposit 100 stETH to mint x senior(stETH) and 100 - x junior(stETH) and swap junior to senior tokens, use the senior tokens as collateral to borrow junior tokens, and repeat the swap procedure. This will put alice in a short position,

- If alice wants leveraged yield to a yield source, she can deposit 100 stETH and mint x senior(stETH) and 100 - x junior(stETH) and swap senior to junior tokens. In addition, she can use the junior tokens as collateral and borrow ETH, mint more stETH, and repeat the process.

- If Alice wants fixed yield to a yield source, she can deposit 100 stETH and mint x senior(stETH) and 100 - x junior(stETH) and swap junior to senior tokens. She will then earn the fixed yield the system is preset to give out to senior holders.

- If Alice wants to generate more yield on top of stETH, she can deposit 100 stETH and mint x senior(stETH) and 100 - x junior(stETH), and provide liquidity to an AMM, such that the aforementioned needs can be satisfied by other agents.

Although the examples are outlined with stETH, note that the vault to be deposited and split can be composed of either a singleton or a combination of vaults, as long as the vaults share ETH as the same underlying.

7.2 Synthetic Assets

We outline an implementation where a `senior` represents a cash secured put position of the asset(a stable unit of account denominated in underlying until some strike price $w * inceptionprice$).

Under the construction where the `tVault` wraps an asset that perpetually realizes volatility(preferrably with a large market cap) such as ETH and its underlying set as USD or a CPI, a senior token with promised return R_f set close to 1 would exhibit the qualities of a stablecoin. It's holders' counterparties would be agents with demand for leveraged ETH exposure, which is assumed to be ample given the highly liquid nature of ETH derivative markets.

This stablecoin formation can be generalized. Under the construction where the `tVault` wraps an asset that realizes volatility and its underlying set as an arbitrary `underlying`(for which an oracle for the asset/`underlying` pair exists), a senior token with promised return R_f set (near) 1 would exhibit the qualities of a synthetic asset of `underlying`. It's instantiation would be identical to that of stablecoins, except the `underlying` need not be restricted to USD, but could be any assets such as commodities, equities, etc. As is that of stablecoins, these instruments would be capital efficient to mint as opposed to existing debt-based models.

A particularly interesting application would be a CPI or a off-chain bond synthetic instrument. Existing implementations that aim to create assets pegged to a CPI or any external yield denominated in USD either require defi rates to lower bound the yield it tracks, or are debt-based. These mechanisms are flawed as it is unreasonable to assume that defi yield would be greater than inflation, or in the case of debt based instruments its demand would be even greater than dollar-pegged stablecoin during bearish conditions. If the asset to be split is a staked representation of an L1/L2 token, such as staked ETH, these staking rewards can be redirected to increase R_f and users would essentially gain network staking rewards on their USD without being exposed to market beta.

As most decentralized synthetic instruments(where the collateral is censorship resistant) are debt-based, under some market conditions its supply does not scale in proportion to its demand. The senior token under this construction, however, is as scalable as fiat-backed stablecoins as it merely resembles a swap between ETH and USD for its minter(as does fiat backed stablecoins). The indirect arbitrage and bounded trade mechanisms will ensure that these senior holders will always mint and redeem at $(R_f)^t$ USD.

An observation to be made is that, as does debt based models, this new stablecoin scales with leverage demand in an aggregate sense. However, this model allows this leverage to be more easily transferrable, as people can hedge via external markets or trade them in exogenous secondary markets.

7.2.1 Dynamic Adjustment

However, under its simplest instantiation, when the value of ETH drops to w of the price at which the tranche pair is initiated, the junior token would be set worthless by the pricing rule(and below w the system would be insolvent when all `seniors` are priced at 1 USD). In contrast when the price of ETH increases much higher than its price during the tranche pair's inception, the supply of senior tokens would be artificially capped by w percentage of the *market cap of ETH when prices were much lower*, limiting its scalability as compared to the current market cap of ETH. It is thus necessary to equip the system with a mechanism that dynamically adjusts the ratio w in accordance to the price fluctuations of ETH.

The dynamic adjustment mechanism works as follows: If the price of ETH increases significantly, the system would a) devalue junior tokens, b) mint more senior tokens(which alters w) and c) compensate the junior token holders with these newly minted senior tokens. This is equivalent to the process where `junior` holders *sell* a portion of their leveraged exposure for `underlying`(USD) for profit.

If price of ETH decreases significantly, the system would a) devalue senior tokens, b) mint more junior tokens(which alters w) and c) compensate the senior token holders with these newly minted junior tokens This is equivalent to the process where `senior` holders *buy* with a portion of their `underlying`(USD) for ETH.

To illustrate how this would be implemented, lets assume that at tranche inception the price of ETH/USD is 1000, and the initial ratio w is set as 0.8, which implies that 1 ETH would split into 0.2 `senior` and 0.8 `junior` tokens.

0.2 `senior` would be initially collateralized by $1000 * 0.2 = 200USD$ and 0.8 `junior` would be collateralized by $1000*0.8 = 800USD$. Note that $P_{su} = P_{ju} = 1000USD$ at this point.

When ETH price increases to 2000, and assuming the senior tokens maintain their price, 0.8`junior` would be collateralized by $2000-200 = 1800$ USD. $P_{su} = 1000USD$ but $P_{ju} = 1800/0.8 = 2250USD$. As it is now twice as less capital efficient to mint one `senior`, the system can decide to devalue `junior` and mint more `senior` such that the `junior` holders can claim the minted `senior` pro rata. For example, the system could mint 0.2 `seniors` for every existing 0.2 `seniors`, such that the total supply of `seniors`doubles and 1 ETH would split into 0.4 `senior` and 0.8 `junior` tokens. As 1ETH is 2000 USD, now 0.8 `junior` tokens are collateralized by $2000-400 = 1600USD$ and the the resulting devalued $P_{ju} = 1600/0.8 = 2000USD$. These newly minted `senior` would be claimable by `junior` holders.

In contrast, when ETH price decreases to 500, and assuming the senior tokens maintain their price, 0.8`junior` would be collateralized by $500 - 200 = 300$ USD, and $P_{ju} = 300/0.8 = 375USD$. Since `junior` holders are exposed to too much leverage the system decides to mint more `junior` and devalue `senior` tokens. For example, if the system mints 0.4`junior` for every circulating 0.8`junior`, then under this new ratio(1 ETH splits to 1.2 `junior` and 0.2 `senior`) 0.2 `senior` is backed by $500 - 1.2 * 375 = 50USD$, and the resulting $P_{su} = 50/0.2 = 250$. These newly minted `junior` would be claimable by `senior` holders.

7.2.2 Resemblance of a cash secured put

An interesting observation to note is that the junior token holders leverages the senior token's share of the `tVault`. It is only natural for the lenders(seniors) to be compensated with the time value of money in the form of interest paid by the juniors. Conveniently, this interest can simply take the form of setting R_f to be a value greater than 1. Denote this interest to be earned from lending as R_{f_I}. R_{f_I} could be a parameter that is controllable; when demand for of `senior` is low(from high leverage demand), R_{f_I} could increase, and vice versa.

Recall that the Dynamic Adjustment process at which, when price drop to a certain level P, senior token holders are compensated with newly minted junior token while their senior tokens are simultaenously being devauled. This is equivalent to the process where the senior token holders are purchasing ETH at P. Thus, this interest bearing senior token resembles an OTM cash secured put position, as they are essentially yield bearing IOU tokens for bids.(in OTM cash secured put sells, a trader provides cash as collateral and agrees to purchase the asset when the price decreases to the strike price)

Moreover, this instantiation gives rise to another yield source. The ETH that was provided as collateral to be split could be staked, where its rewards would then be distributed to the senior and junior holders. This effectively allows the senior holders to be exposed to ETH staking rewards without being exposed to the volatility of ETH. This is implemented simply by further increasing R_f.

Denote the `senior` yield to be generated from the staking rewards as R_{f_S}, then the system would set

$$R_f = R_{f_S} + R_{f_I}$$

R_f would then be a relatively risk free source of yield for this new stablecoin backed with censorship resistant collateral. R_{f_I} would be covered by the `junior`'s share of the staking reward, and thus the `junior` token holders would leverage without cost.

7.3 Comparison to debt-based models

We outline some advantages over debt-based synthethic asset models.

1. Our system does not require an external party of liquidators, where the system necessitates incentives in the form of premiums. This serves to be wasteful value leaked from the minters. In addition, as price feeds update frequency are bounded by block times(and is often further delayed to prevent manipulations) the auction might fail when the collateral/synthethic asset pair is highly volatile.

2. Swapping is a 1:1 value transfer mechanism and is more capital efficient than 1+x:1 borrowing.

3. Instead of being forced to *sell*, the minter is forced to *buy* the asset at low prices

7.3.1 Expansion and Contraction

The split and swap (or swap and merge) procedure can be abstracted to a minting(redeeming) module when there is infinite liquidity in the AMM for the given value P_{js}. It is, however, unreasonable to assume the exact amount of liquidity the AMM would have at any given time and value P_{js}, and the liquidity of the pair in the AMM would serve as the bottleneck for efficient minting and redeeming.

For every 1 `senior` "minted", the program necessitates $\frac{1-w}{w}$ `junior` to be "minted" as well. This would require a counterparty that is willing to take a margin long on ETH(when assuming the asset `tVault` is ETH). The source of this counterparty can arise from either an organic demand for leveraged longs and if those aren't sufficient it could be replaced by arbitrageurs who can hedge the same notional exposure in an external derivative market.

Similarly, a contraction would occur when the users "redeem" by swapping a portion back to `junior` and merging. This would require existing `junior` holders to close their leverage longs or require market makers to perform the operations outlined in the indirect arbitrage section.