Multiply: Tokens Whose Fees Run a Leveraged Position

multiply.cash
Robinhood Chain

August 2026

Abstract

The system described here is a token launchpad on Robinhood Chain in which every launched token permanently locks the entirety of its liquidity and streams its trading fees into a live leveraged perpetual position on Lighter, an onchain order-book exchange. The token itself is deliberately inert: fixed supply, no reachable administrative authority, no transfer tax. The engine sits outside it. The pool's swap fee, 1% to 5% as set at launch, is converted to USDG inside each swap and delivered to a per-token contract that forwards 70% of it to a per-token treasury, which posts it as collateral on a perpetual market chosen at launch — any Lighter market, long or short, at 2× to 50× leverage. Profit is realized tranche by tranche under a fixed take-profit policy; the capital of a realized tranche and 70% of its gain buy back and burn the token, and the remaining 30% of the gain accrues to the operator, together with 25% of trading fees, as the protocol's only revenue. This paper specifies the launch mechanics, the fee circuit, the tranche accounting, and the trust model, and states the properties and risks that follow from the design.

1. Introduction

Token launchpads have converged on a common design. A token with fixed supply is sold through a curve or seeded into a pool, liquidity is locked, and the trading fees the token generates are treated as income: split among the creator, the protocol, and sometimes the pool itself. The fee stream is distributed, not deployed.

It is also all there is. A token that nobody trades earns nothing, and nothing happens to it; the markets around it keep moving all the same. This paper specifies a launchpad in which the fee stream is working capital, and so keeps working when trading stops. Each token launches with its full supply seeded one-sided into a Uniswap v4 pool quoted in USDG [3], built and permanently locked by the Doppler contracts [4]. The pool's fee leaves every swap as USDG, and most of it reaches a treasury that exists for that token alone. The treasury posts it as collateral on a leveraged perpetual — market, direction, and leverage chosen by the creator at launch — executed on Lighter's Robinhood deployment [1]. When a tranche of that position reaches its target it leaves the engine whole, and what it brings back buys the token on its own pool and burns it.

The design differs from reserve-coupled systems, in which a tokenized leveraged position is the reserve asset of the curve and the token's price mechanically tracks the underlying. Here the coupling is deliberately loose. The token trades on an ordinary spot pool; the perpetual affects its price only through realized flows — buybacks — and through supply reduction. Two consequences follow. First, failure of the position cannot cascade into the pool: a liquidation costs the engine its posted collateral and nothing else, and the engine restarts from the next fees collected. Second, exposure to the underlying is realized rather than marked: holders receive it as burns, not as a continuously repricing reserve.

Sections 2–5 specify the mechanics. Section 6 states the properties the mechanics produce, including an explicit trust model. Section 7 states the risks.

2. Launch Mechanics

A launch is one transaction to a router, which assembles the market on Doppler's Airlock [4] with the shape every token shares and records the token's engine. The router holds no funds, and has no function that reaches a token or its pool afterwards.

Token. A standard ERC-20 with fixed supply $S = 10^9$ units, minted once. There is no minting after launch, no transfer tax, no pause, and no blacklist. The owner is the Airlock contract, whose paths to the owner's functions are closed for a locked pool; no person holds that authority. Any holder may burn its own balance, and a burn reduces total supply. The quantity reported as burned is the identity $B(t) = S - \text{supply}(t)$, recomputable by anyone from the chain.

One-sided seeding. At launch the entire supply is placed as one-sided liquidity in a Uniswap v4 pool against USDG [5]. The opening price p_0 is set from a target valuation V_0 (currently about $4,000):

$$p_0 = V_0 / S, \quad \text{so} \quad MC_0 = p_0 \cdot S = V_0$$

Every token opens at the same dollar valuation regardless of market or leverage choice. There is no graduation event: the pool seeded at launch is the token's market for life. Purchases walk up the ask ladder of the locked position; there is no migration, no LP token, and no moment at which liquidity changes hands.

The lock. The liquidity belongs to Doppler's initializer, the hook of the pool. A pool created with fee beneficiaries is *locked*: it has no exit, no withdrawal, and no migration. The functions that could alter a live pool are gated on the token's governance timelock, and every token is launched with the no-op governance, whose timelock is the burn address.

The fee. The pool's LP fee is zero. A second hook takes the trading fee φ, between 1% and 5%, chosen at launch and immutable, on the output of each swap. A fee taken in the token is swapped into USDG inside the same call, so what leaves the pool is USDG only and no inventory of tokens waits to be sold.

Launch protection. The fee may open at $\varphi_0 \le 80\%$ and fall in a straight line to φ over a window $w \le 600$ s:

$$\varphi(t) = \varphi_0 - (\varphi_0 - \varphi)\,\min(1,\, t / w)$$

A purchase in the launch block keeps little of what it buys, and the difference loads the engine. An optional first buy by the launcher is bundled into the launch: it is the pool's first swap by construction.

The sink. Each launch creates a contract for that token alone, the fixed destination of its fees. Its operational function, *flush*, is permissionless, and its only reachable outcome is to send the operator's share and the engine's share to their recipients and to burn any of the token it holds. The shares are written at creation and have no setter.

Per-token treasuries. Each token's treasury address is derived deterministically from an operator master secret k as HMAC-SHA256(k, $\text{addr}_{\text{token}}$) [6]. The private key exists only inside the keeper process; only the address is ever persisted. The treasury receives the engine's share from the sink and owns the token's account on Lighter, and the venue accepts orders for an account only from its owner's key. Tokens are therefore pairwise isolated — one token's funds can never touch another token's position — and the creator, who never holds any of these keys, cannot drain or borrow against the position backing their own token.

The engine. Market, side σ ($+1$ long, -1 short), leverage λ and take-profit r are stored on the router. A *fixed* token keeps its market and side for life. A *managed* token's creator may change all four, each change announced onchain and effective only after a notice common to every token (currently one hour). The router's owner may override leverage and take-profit of any token, for a venue that lowers a limit, and never its market or side.

3. The Fee Circuit

Every trading fee, in USDG, is divided by a vector β over (engine, operator, Doppler):

$$\beta = (0.70,\ 0.25,\ 0.05)$$

Doppler's share is the protocol fee of the launch contracts. The division between engine and operator is enforced by the sink, not by the keeper. The creator's share is zero — the creator's return is the token itself.

An off-chain keeper passes over every token continuously, pausing 15 seconds between passes. For each token it credits what has come home from the venue, flushes the sink when it holds more than a dust floor, runs the buyback of Section 5, and operates the position: deposit, withdrawal, take-profit, opening.

Deposits. The perpetual opens once the fees waiting reach a gate $G = \$25$. The first position waits ten minutes from the launch, so that the fees of the opening enter together; while a position is open, the fees accrued since the last tranche, again at least G, enter as one tranche an hour at most; after a take-profit has emptied the position the next opens as soon as the gate is met. A deposit is a plain USDG transfer to a deterministic intent address, which credits the token's own Lighter account across the bridge. Positions run isolated margin. Collateral c at leverage $\lambda \in \{2, 3, 5, 10, 20, 25, 50\}$, never above the market's own limit, extends the position by notional

$$N = \lambda \cdot c$$

executed as an immediate-or-cancel order within 2% of the mark. Only cash is deployed: profit not yet realized, which the venue counts as available, buys nothing.

Crash safety. Every payment, order, and withdrawal is recorded on disk before it is sent. After a crash or a timeout an order is settled on the trades it actually made, found by its own identifier, and a withdrawal on whether its cash has left the account; until the outcome is known nothing else moves for that token. An onchain spend is decremented before it is sent, so that a crash under-pays a bucket for one pass rather than paying it twice.

4. Tranche Accounting

The venue nets all of a token's deposits into a single position. The keeper, however, keeps its own decomposition: every opening creates a *tranche* $(c_i,\ m_i,\ q_i)$ recording its collateral, the price it filled at, and its size $q_i = \lambda c_i / m_i$.

Take-profit rule. A token's take-profit r is chosen at launch between 0.10 and 0.20 λ, and never above 5. Tranche i closes when the mark reaches

$$m_i^* = m_i\,(1 + \sigma\,\frac{r}{\lambda})$$

Since the tranche's profit at mark m is $q_i\sigma(m - m_i) = \lambda c_i \sigma(m - m_i) \,/\, m_i$, reaching m_i^* is worth exactly $r \cdot c_i$: a move of r/λ in the underlying is a return of $+r$ on that tranche's own collateral. The upper bound on r keeps that move within 20%, so that a target can be reached.

No dilution from later deposits. Under averaged-entry targets, a later deposit at a worse price shifts the aggregate break-even and postpones profit-taking on capital that was already in profit. Per-tranche targets are invariant to later deposits: every dollar of fees runs toward its own target from the price at which it entered. Figure 1 shows the required underlying move r/λ across the leverage range.

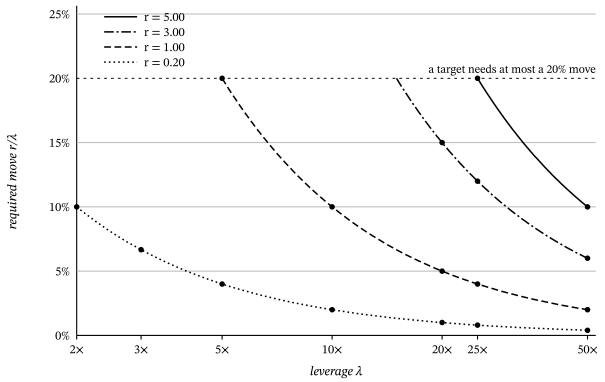

Figure 1: The underlying move r/λ required to close a tranche, as a function of leverage (logarithmic axis), for four take-profits. Dots mark the leverages offered at launch. A 5× token at +20% closes each tranche on a 4% move of the underlying; a 50× token at +20% on 0.4%. No target may require more than 20%.

Decay. A distant target can leave a tranche in profit and never closed. A tranche of age a (in days) runs to

$$r(a) = r - (r - 0.10)\,\mathrm{clamp}_{[0,1]}\,\frac{a - 7}{23}$$

that is, unchanged for a week and then falling in a straight line to +10% on day 30. The keeper never closes a tranche at a loss on its own; one that is under water waits.

Proceeds. A tranche decides when and how much to close; what the close pays is what the venue pays. For a position of size Q, average entry $\bar{m}$ and margin A, closing a size q at an average fill p takes out of the engine a capital K and a gain g

$$\pi = \sigma q\,(p - \bar{m}), \quad g = \pi^+, \quad K = (A\,\frac{q}{Q} - \pi^-)^+$$

where x^+ and x^- are the positive and negative parts. Funding is settled by the venue in the margin A, and is therefore inside K. With a single tranche open π is its own profit. With several, the difference between a tranche's entry and the average stays in the position and comes out with the tranches that are left: over a position that is closed entirely the sums coincide.

Execution. When a tranche reaches its target, the keeper closes exactly that tranche's size with a reduce-only immediate-or-cancel order under the 2% slippage cap. The fill is read from the order's own trades; a partial fill leaves the remainder queued, and an order refused by a thin book is retried at half the size.

Reconciliation. The venue is authoritative. If the position is gone — liquidation — the tranche book zeroes. If the venue's size is smaller than the book, tranches are rescaled pro-rata; if larger, the excess becomes a synthetic tranche at the current mark. The keeper's accounting can only ever converge toward what the venue reports.

Ring-fencing. What a take-profit took out, capital and gain, is excluded from the balance the keeper may deploy (deployable = cash $- K - g$). It can only leave toward the buyback; the next position is loaded by new fees.

Change of engine. When a managed token's new market or side takes effect, the whole position is closed at market, reduce-only, and nothing opens until that close has filled. A net gain is banked like any other; a net loss is final; what is left is the collateral of the new position.

5. Profit Return and Supply Dynamics

Withdrawal. Once what take-profits took out clears a floor W = \$20, it is withdrawn from Lighter over the zk bridge (up to about 30 minutes) back to the token's treasury. The keeper recognizes the settlement from the on-chain transfer itself and divides it as

$$\text{buyback} = K + 0.70\,g, \quad \text{operator} = 0.30\,g$$

The capital has already been divided once, as a fee, and is not divided again: a dollar that enters the engine leaves it either through a liquidation, or whole, and with it the larger part of what it earned.

Buyback. When a token's buyback reserve crosses \$20, the keeper buys the token on its own pool and burns what it bought. A single buy spends at most \$25, with a minimum output quoted from live pool state less 3%, so a large reserve drains as a sequence of ordinary buys rather than one block-moving order.

An example. A tranche that closes on target at $r = 0.50$ returns 1.5 times its capital: 1.35 times it buys the token and burns it, 0.15 times it goes to the operator.

Supply identity. Reported burns aggregate two channels in one number, $B(t) = S - \text{supply}(t)$: the buyback burns funded by take-profits, with the little of the token that reaches a sink or a treasury, and any voluntary burn by a holder. Trading fees themselves are not burned: they are put to work, which is what makes the buyback larger than they were.

Operator revenue. The operator earns 25% of trading fees and 30% of realized gains, and nothing else: no launch fee, no share of capital, no share of the pool. Of those earnings, 70% buys the platform token, MULTIPLY (`0xBe021bF099efF1C81d799079358943A981b77642` on Robinhood Chain), on its own pool and burns it; 30% stays in the operator's treasury. The keeper counts them from the transfers of the sinks and the treasuries alone, every five minutes, and buys in a size that moves the price by at most 1%.

6. Properties

Fee continuity without the operator

The conversion of fees to USDG happens inside the swap, and *flush* is permissionless with fixed shares. Both continue to function with the keeper offline, operated by anyone willing to pay gas.

Bounded engine failure

The perpetual runs isolated margin, so the worst case of the engine is the loss of its posted collateral. The pool, the locked liquidity, and holder balances are outside the blast radius of a liquidation; what take-profits have already taken out is outside it too, and the engine re-opens at the next G of fees.

Monotone extraction

Capital that has taken profit is never at risk again. The engine cannot give back a gain it has realized, because neither the gain nor the capital that made it re-enters a position.

Constant opening valuation

$\text{MC}_0 = V_0$ for every launch, regardless of the market, direction, or leverage chosen.

Per-token isolation

Each token has a distinct sink, a distinct treasury address, and a distinct venue account, and the venue only accepts orders from an account's owner. No token's flows can touch another token's position, and no creator holds a key to any of it.

Realized, not marked, exposure

The token's spot price does not mechanically track the underlying. Exposure reaches holders only through realized flows: take-profits that become buybacks and burns. This is the price of the bounded-failure property above, and it is deliberate.

Verifiability

Every launch, fee, flush, burn, deposit, and buyback is a confirmed transaction on Robinhood Chain [2], every engine and every change to it is a public event, and each token's sink and treasury are single public addresses. Quote inflow equals sink transfers plus perpetual withdrawals; outflow equals venue deposits, buyback swaps, and the operator's share of gains. The accounting closes, publicly.

Trust model

The design is explicit about what is trustless and what is not. Trustless: the token's supply and the absence of reachable authority over it; the permanence of the liquidity lock; the fee rate; the division of fees between engine and operator. Trusted: the keeper's execution of deposit, take-profit, and buyback policy; custody of the master secret from which treasuries derive; the division of gains; the router's owner, who sets the terms of future launches, may override leverage and take-profit, may re-point a sink whose treasury became unusable, and receives the operator's share. The operator can stall the engine or mismanage it; the operator cannot unlock liquidity, mint, or change a token's fee, because no contract has a function for any of them.

7. Risks

Liquidation

The venue closes an isolated position while some margin is left: at a maintenance fraction μ of notional that depends on the market (today 1.2% for BTC, 3% for NVDA). The adverse move that liquidates is therefore not $1/\lambda$ but

$$d = \frac{1}{\lambda} - \mu$$

At 50× on BTC that is 0.8%; at 25×, 2.8%; at 10×, 8.8%. Liquidation forfeits the collateral in the position and zeroes the tranche book; the engine restarts from the next $25 of fees. Higher leverage produces faster burns on a good call and faster liquidation on a bad one, and near a market's limit the target can be further than the liquidation.

Idle engines

After a take-profit a token holds no position until its fees reach the gate again. A token nobody trades keeps the po-

sition it has, and does not open another.

Managed tokens

A managed token's creator chooses what it trades. Each change costs a close and an opening, and on a thin market that can be a large part of the collateral. The notice lets holders see it coming; it does not prevent it.

Operator dependency

Deposits, take-profits, withdrawals, and buybacks are executed by a single off-chain keeper. If it halts, those flows pause — fees keep accruing in USDG — until it resumes. More severely, treasury keys derive from the operator's master secret: its loss would strand treasury balances and venue collateral, and its theft would expose them, though never the locked liquidity or holder balances.

Venue and bridge dependency

Lighter, its zk bridge, Doppler's contracts, and the chain's RPC infrastructure are live dependencies. Outages pause deposits, closes, or withdrawals until they recover; bridge settlement adds latency (minutes, not blocks) to every return. Markets with trading hours neither open nor close a position outside them.

Early liquidity depth

A token opens at a $4,000 valuation with one-sided liquidity, so early trades have high price impact in both directions. The keeper's own buybacks are bounded by the $25 cap and the 3% slippage floor precisely so that the engine never trades meaningfully against that thinness.

Indirect exposure only

A buyer seeking continuous delta exposure to an underlying should hold the perpetual itself. The token delivers that exposure only as realized flows, on the keeper's schedule, and a position that never reaches its targets delivers none of it.

8. Conclusion

This paper specifies a launchpad in which a token's trading fees are not income to be split but collateral to be deployed. The token is inert; its liquidity is locked for good; its fees arrive as USDG and run a live leveraged perpetual, and what that position wins leaves it whole, to buy the token back and burn it. What is enforced by contract — supply, lock, fee, and its division — survives everyone, including the operator. What is operated — the engine — is bounded in what it can lose and public in everything it does.

References

[1] Lighter, "Lighter Documentation," https://docs.lighter.xyz

[2] Robinhood Chain explorer (Etherscan), https://robin.etherscan.io

[3] H. Adams, M. Salem, N. Zinsmeister, S. Reynolds, A. Adams, W. Pote, M. Toda, A. Henshaw, E. Williams, D. Robinson, "Uniswap v4 Core," 2023.

[4] Whetstone Research, "Doppler Documentation," https://docs.doppler.lol

[5] Global Dollar Network, "USDG — Global Dollar," https://globaldollar.com

[6] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, February 1997.