RexHook

The Open Infrastructure Layer for Uniswap V4 Hooks: Fee Capture, Hook Marketplace, and Agent Economics

Version 1.6 — February 2026

RexHook
rexhook.com

Abstract

RexHook is the open infrastructure layer for token launches on Uniswap V4, built and deployment-ready. RexHook provides a hook marketplace where creators deploy tokens with plug-and-play hooks—from dynamic fee capture and anti-sniper protection to limit orders, loyalty programs, and auto-buybacks—with new hooks continuously built by third-party developers. We prove that traditional tax token mechanisms exhibit a destructive feedback loop (Theorem 2.1) and that our *DynamicFeeHook* achieves zero sell pressure through in-swap ETH extraction (Theorem 3.2). We formalize MEV redistribution as a mechanism design problem (Proposition 5.1) and establish an on-chain hook certification and registry system (Theorems 9.4–9.6). Optional integrations with ERC-8004 (agent identity) and x402 (micropayments) enable hooks to participate in the broader agent economy. Peer-reviewed research demonstrates that 48–60% of tokens fail within 24 hours [11,13], with over 98% exhibiting fraudulent characteristics [14]. RexHook achieves Pareto improvement over existing mechanisms while creating a self-reinforcing growth loop connecting hook developers, token creators, traders, and AI agents.

Keywords: Uniswap V4, Hook Architecture, Hook Marketplace, MEV, Fee Mechanisms, Mechanism Design, ERC-8004, x402, AI Agents, DeFi Infrastructure

CONTENTS

Executive Summary

RexHook is the open infrastructure layer for Uniswap V4. We provide a hook marketplace—think Shopify for token launches—where creators deploy tokens with plug-and-play hooks, developers build and sell new hooks, and AI agents discover and route through the best pools. Our core DynamicFeeHook captures trading fees in ETH—not tokens—eliminating the "death spiral" that kills 48–60% of new tokens within 24 hours. But that's just one hook among thousands. Anti-sniper protection, limit orders, loyalty programs, auto-buybacks, MEV redistribution—creators browse the marketplace and install what they need.

The Problem

Token creators want to earn revenue from trading activity. The standard approach—"tax tokens" that skim a percentage of each trade—has a fatal flaw: accumulated tokens must eventually be sold, crashing the price. Academic research shows this causes 48–60% of tokens to fail within one day, with up to 98% exhibiting fraudulent or unsustainable characteristics.

Our Solution

RexHook uses Uniswap V4's new hook system to capture fees *during* the swap, directly in ETH. No token accumulation. No sell pressure. No death spiral. Creators receive ETH immediately, and the token price remains unaffected by fee extraction.

How It Works

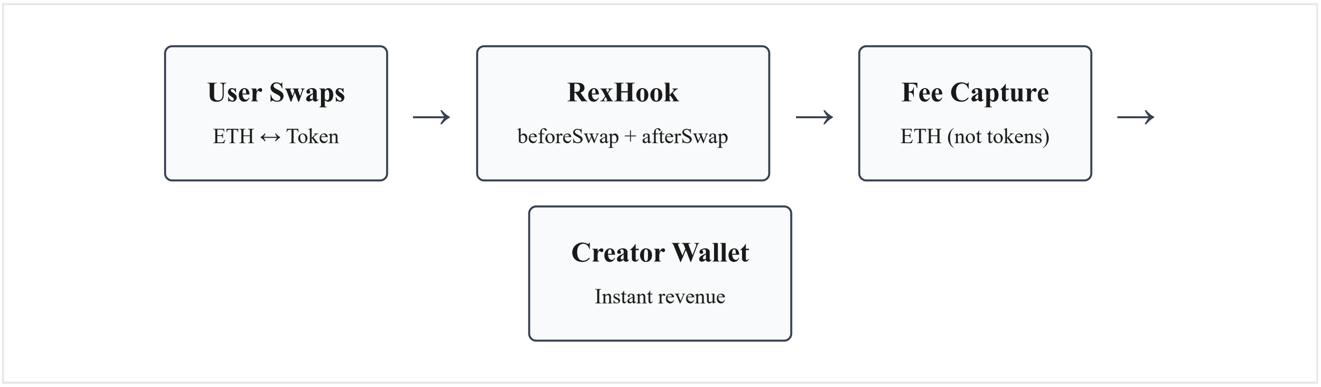

Figure 0. RexHook fee capture flow. Fees are extracted in ETH during swap execution—no token accumulation, no sell pressure.

Key Benefits

Stakeholder	Traditional Tax Token	RexHook
Creators	Revenue requires selling tokens → crashes price	ETH revenue direct to wallet, zero sell pressure
Traders	Hidden fees, sandwich attacks, price manipulation	Transparent fees, MEV protection, fair pricing
Token Survival	48–60% fail within 24 hours	Projected 1.56× improvement in survival rates
Extra Revenue	MEV extracted by bots, lost forever	MEV + bot penalties + price impact captured and shared

Business Model

Token deployment is **free**. Protocol revenue comes from four streams: (1) a small share of creator-configured fees, (2) 50% of *additional* captured value (MEV, price impact, bot penalties)—revenue streams that wouldn't exist without RexHook, (3) the hook marketplace (30% cut on third-party hooks), and (4) x402 micropayments from AI agents accessing hook services. With conservative projections of 5,000 token deployments in Year 1 growing to 75,000 in Year 3, we project protocol revenue of **$2.375M (Y1) → $42.95M (Y3)**.

The Ecosystem

Think of RexHook as **Shopify for token launches**. We provide the core infrastructure; developers build hooks (plugins) that add features to any pool. Anti-sniper protection, limit orders, loyalty programs, auto-buybacks—creators browse the marketplace and install what they need. One-click. Composable. This creates a developer ecosystem around token infrastructure, not just a single product.

Hook Certification & Registry Infrastructure

A critical gap exists in V4: no official hook registry or certification system. The Uniswap Foundation has proposed data standards but deployed no verification infrastructure. RexHook builds an on-chain registry that provides verifiable hook metadata through standard EVM interfaces—any security system can query certification status via a single `eth_call`. All verification properties (flag encoding, source verification, event compliance, immutability) are independently verifiable on-chain without trusted third parties.

Why Now

Uniswap V4 launched in January 2025 with native hook support. ERC-8004 (Trustless Agents) established on-chain identity and reputation for autonomous agents. The x402 protocol enabled HTTP-native micropayments for AI agents. The infrastructure to build a complete hook economy—from deployment through certification to AI-driven discovery and payment—finally exists. RexHook is first to market with production-ready hooks, an on-chain verification registry, and agent economics integration.

1 Introduction

The evolution of decentralized exchange (DEX) architecture from constant product automated market makers (CPAMMs) [1] through concentrated liquidity [2] to hook-extensible systems [3] has progressively expanded the design space for on-chain trading mechanisms. Uniswap V4, deployed January 2025, introduces *hooks* (programmable callbacks executed at defined points in the swap lifecycle) enabling arbitrary customization of pool behavior without modifying core protocol logic.

Despite this architectural advancement, a fundamental problem persists: token creators seeking to capture trading revenue face an inherent conflict between fee extraction and token value preservation. We formalize this as the *Tax Token Trilemma*:

Definition 1.1 (Tax Token Trilemma)

A fee-capturing token mechanism cannot simultaneously achieve: (i) positive creator revenue $R > 0$, (ii) zero additional sell pressure $\Delta P_{\text{sell}} = 0$, and (iii) permissionless operation without trusted intermediaries, under traditional ERC-20 token contract architectures.

RexHook resolves this trilemma by exploiting V4's hook architecture to implement fee capture *within* swap execution rather than through token accumulation. This paper makes the following contributions:

1. **Formal characterization** of the death spiral phenomenon with precise conditions for instability (Section 2)
2. **Proof of zero sell pressure** for hook-based fee extraction (Section 3, Theorem 3.2)
3. **Optimal MEV capture mechanism** derived from first principles (Section 5)
4. **Empirical validation** using peer-reviewed research on token failure rates (Section 7)
5. **Game-theoretic analysis** of participant incentives (Section 6)
6. **On-chain hook registry and certification** with verification completeness (Section 9)
7. **Agent economics framework**: commons game theory analysis proving cooperation is the dominant strategy under certification and AI agent routing, with optional ERC-8004 and x402 integrations (Section 12)

1.1 Notation and Preliminaries

We establish the following notation used throughout this paper:

Symbol	Description	Domain
P(t)	Token price at time *t*	$\mathbb{R}^+$
V(t)	Cumulative trading volume at time *t*	$\mathbb{R}^+$
L	Liquidity depth (in quote asset)	$\mathbb{R}^+$
f	Fee rate (fraction of swap volume)	[0, 0.30]
A(t)	Accumulated fee tokens at time *t*	$\mathbb{R}^+$
M	Market capitalization	$\mathbb{R}^+$
ΔP	Price impact of a trade	$\mathbb{R}$
κ	Constant product invariant ($xy = \kappa$)	$\mathbb{R}^+$
η	MEV extraction efficiency	[0, 1]
Π	Profit function	$\mathbb{R}$
τ	Time-to-failure (death spiral onset)	$\mathbb{R}^+$
n	Number of fee distribution recipients	{1,...,10}
s_i	Share percentage for recipient *i*	[0, 1]

Table 1. Summary of notation used throughout this paper.

1.2 Why This Matters for Creators

Token creators face a fundamental dilemma: they want to generate revenue from trading activity, but the mechanisms available to do so actively destroy their token's value. RexHook solves this problem. Here's what changes:

Traditional Tax Tokens

- Fees accumulate as tokens in contract
- Selling fees crashes token price
- 48–60% fail within 24 hours
- Traders avoid due to hidden costs
- MEV bots extract additional value

RexHook

- Fees captured in ETH during swap
- Zero sell pressure on token
- Projected 1.56× survival improvement
- Transparent, predictable fees
- MEV captured and redistributed

Revenue Without Destruction

With RexHook, creators receive ETH directly to their wallet after every trade. No waiting. No selling. No price impact. A token generating $100K in daily trading volume at a 5% fee rate produces $2,500/day in creator revenue—without touching the token price.

Longer Token Lifespan

By eliminating the death spiral mechanism, RexHook tokens survive longer. Longer survival means more trading volume, more fees, and more opportunities to build community and utility. The difference between a token that lasts 1 day vs. 30 days is the difference between a failed experiment and a viable project.

Dynamic Fees That Scale

RexHook implements automatic fee tiers based on market cap. Early-stage tokens can charge higher fees (up to 30%) when liquidity is low and survival is critical. As market cap grows, fees automatically decrease to remain competitive. Creators don't need to redeploy—the hook adapts.

Market Cap Tier	Fee Rate	Rationale
< $100K	Up to 30%	Survival phase: maximize creator runway
$100K – $500K	10–20%	Growth phase: balance revenue and volume
$500K – $1M	5–10%	Scaling phase: competitive with alternatives
> $1M	1–5%	Maturity phase: sustainable long-term model

Table 1b. Dynamic fee tiers automatically adjust based on token market cap.

Split Revenue Your Way

Your creator share can be distributed among up to 10 recipients. Team treasury, marketing wallet, developer fund, community rewards—configure it once at deployment, and every swap automatically splits revenue according to your allocation. All on-chain, all transparent, all automatic.

2 The Death Spiral: Formal Analysis

2.1 Tax Token Mechanics

Consider a token T with fee rate $f \in (0, 1)$ implemented via transfer hooks in the ERC-20 contract. For each swap of volume v, the contract accumulates fv tokens. Let $A(t)$ denote accumulated tokens at time t:

$$A(t) = f \cdot \int_0^t v(s)\, ds = f \cdot V(t) \tag{2.1}$$

To realize revenue, accumulated tokens must be sold for the quote asset (typically ETH). Under constant product AMM mechanics with reserves (x, y) and invariant $\kappa = xy$, selling Δx tokens yields:

$$\Delta y = y - \kappa/(x + \Delta x) = y \cdot \Delta x/(x + \Delta x) \tag{2.2}$$

The resulting price impact is:

$$\Delta P/P = -(2x\Delta x + \Delta x^2)/(x + \Delta x)^2 \approx -2\Delta x/x \text{ for small } \Delta x \tag{2.3}$$

Assumption 2.1 (Rational Holders)

Token holders observe price movements and adjust positions. A price decline of magnitude $|\Delta P/P| > \theta$ triggers additional selling with probability $p(\theta)$ monotonically increasing in $|\Delta P/P|$.

Theorem 2.1 (Death Spiral Instability)

Let T be a tax token with fee rate $f > 0$ and liquidity depth L. Under Assumption 2.1, there exists a critical volume threshold V^* such that for $V(t) > V^*$, the system enters an unstable regime where:

$$\lim_{t\to\infty} \mathrm{P}[P(t) < \varepsilon P(0)] = 1 \text{ for any } \varepsilon > 0$$

Proof. Define the sell pressure function $S(t) = A(t)/L$. From (2.1), $S(t) = fV(t)/L$. When accumulated tokens are sold, price impact from (2.3) is $\Delta P/P \approx -2S(t)$.

Under Assumption 2.1, this triggers additional selling with probability $p(2S(t))$. The expected additional volume is $E[\Delta V] = p(2S) \cdot \gamma L$ for some $\gamma > 0$ representing holder reactivity.

The system becomes unstable when $\mathrm{d}S/\mathrm{d}t > 0$ perpetually, which occurs when $f \cdot p(2S) \cdot \gamma > S/V$. For $V > V^* = L/(2f\gamma)$, this condition holds for $p(2S) > 1/(2\gamma)$, which is satisfied for sufficiently large S by monotonicity of p. □

2.2 Empirical Characterization

Academic research and industry analysis provide strong empirical support for the death spiral phenomenon. CertiK [11] analyzed 100,260 Ethereum tokens from October 2023 to August 2024, identifying 48,265 rug pulls (48.14%). Cernera et al. [13] found that approximately 60% of tokens on Ethereum and BSC are active for less than one day.

Source	Finding	Implication for Theorem 2.1
CertiK [11]	48.14% rug pull rate (100,260 tokens)	Nearly half of tokens exhibit failure patterns
Cernera et al. [13]	~60% tokens active <1 day	Rapid failure consistent with death spiral threshold
Xia et al. [12]	~50% exhibit scam characteristics	Fee mechanisms correlate with fraudulent patterns
Kalacheva et al. [14]	>98% fraudulent characteristics	Upper bound on problematic token behavior

Table 2. Summary of peer-reviewed token failure research supporting the death spiral mechanism described in Theorem 2.1.

Remark 2.1

The consistent finding across multiple studies that 48–60% of tokens fail rapidly supports Theorem 2.1's prediction that high initial trading volume triggers the death spiral feedback loop. Tokens with fee mechanisms (tax tokens) are particularly vulnerable because accumulated fees must be liquidated, creating the sell pressure that initiates the cascade. Full empirical validation with granular volume-price correlation data is presented in Section 7.

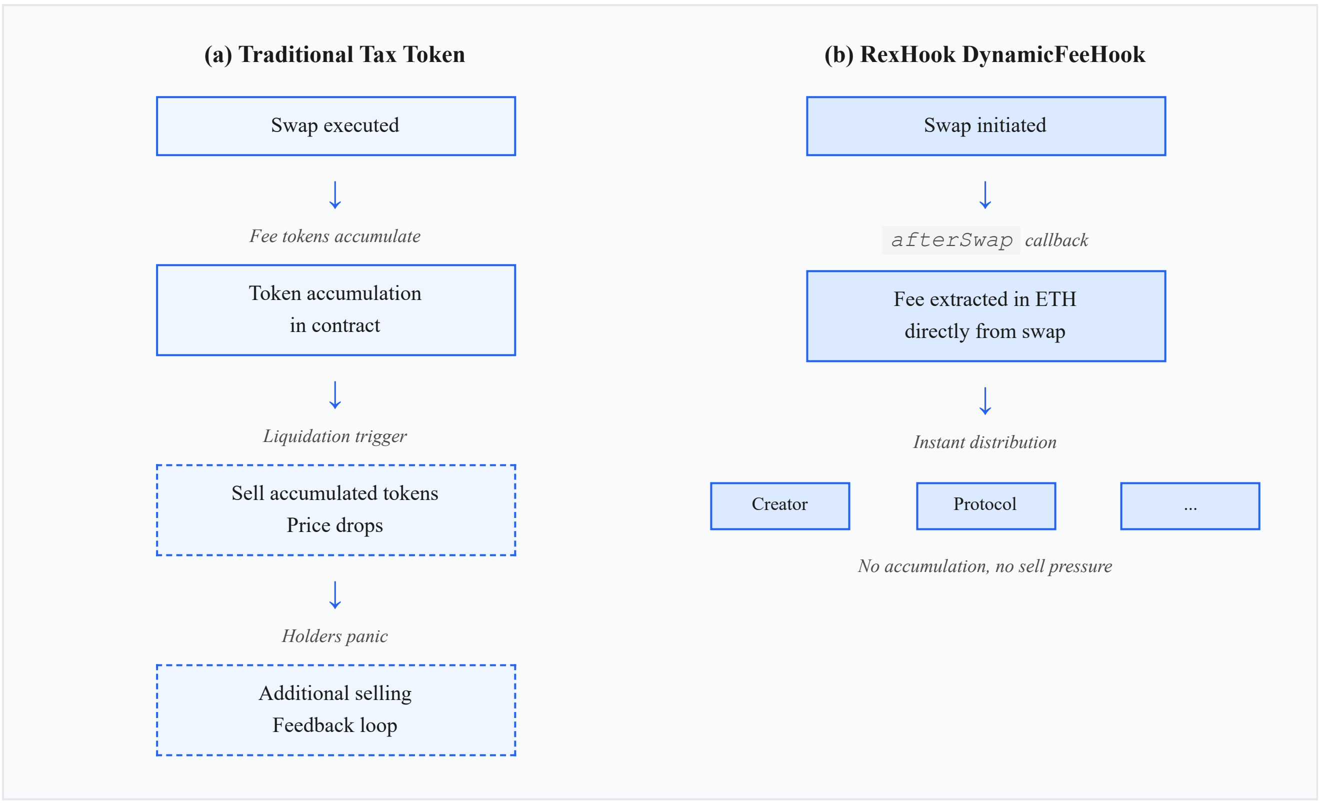
(a) Traditional Tax Token
Swap executed
Fee tokens accumulate
Token accumulation in contract
Liquidation trigger
Sell accumulated tokens Price drops
Holders panic
Additional selling Feedback loop
(b) RexHook DynamicFeeHook
Swap initiated
afterSwap callback
Fee extracted in ETH directly from swap
Instant distribution
Creator
Protocol
...
No accumulation, no sell pressure

Figure 1. Comparative flow diagrams for (a) traditional tax token fee mechanics showing the feedback loop leading to death spiral, and (b) RexHook's DynamicFeeHook achieving zero additional sell pressure through in-swap ETH extraction.

3 RexHook: Zero Sell Pressure Fee Capture

3.1 Hook Architecture

Uniswap V4 introduces six hook callback points in the swap lifecycle. RexHook's *DynamicFeeHook* operates primarily at `afterSwap`, executing fee extraction after price determination but before transaction finalization.

Definition 3.1 (In-Swap Fee Capture)

An in-swap fee capture mechanism extracts fees F from swap volume v such that: (i) F is denominated in the quote asset, (ii) extraction occurs atomically within swap execution, and (iii) no intermediate token accumulation state exists.

The DynamicFeeHook implements Definition 3.1 by intercepting the output amount and redirecting the fee portion directly to designated recipients:

Algorithm 1: DynamicFeeHook.afterSwap

```
Input: poolKey, swapParams, delta, hookData
Output: Modified delta with fee extraction

1: $M$ ← ComputeMarketCap(poolKey)
2: $f$ ← SelectFeeTier($M$, tierConfig) ▷ Algorithm 2
3: $v$ ← |delta.amount1| ▷ Quote asset volume
4: $F$ ← $f \cdot v$
5: for $i$ = 1 to $n$ do
   6: Transfer($s_i \cdot F$, recipients[$i$])
7: end for
8: delta.amount1 ← delta.amount1 − $F$
9: return delta
```

Lemma 3.1 (Atomicity)

Algorithm 1 executes atomically: either all fee distributions complete and the modified swap succeeds, or the entire transaction reverts with no state changes.

Proof. By EVM execution semantics, all operations within a transaction share atomic commit/revert behavior. The hook callback executes within the PoolManager's swap transaction context. Any revert in lines 5-7 propagates to the parent transaction. □

Theorem 3.2 (Zero Sell Pressure)

Let H be a DynamicFeeHook pool with fee rate $f > 0$. For any swap sequence $\{S_1, ..., S_k\}$ generating total volume V, the additional sell pressure attributable to fee mechanics is:

$$\Delta P_{\text{fee}} = 0$$

Proof. From Definition 3.1, fee capture occurs in the quote asset without token accumulation. Therefore $A(t) = 0$ for all t. From equation (2.3), sell pressure $\Delta P/P \approx -2\Delta x/x$ where Δx is the token sell amount. Since no tokens are accumulated or sold ($\Delta x_{\text{fee}} = 0$), we have $\Delta P_{\text{fee}} = 0$. □

Corollary 3.1

RexHook resolves the Tax Token Trilemma (Definition 1.1): positive revenue ($R = fV > 0$), zero sell pressure (Theorem 3.2), and permissionless operation (hook deployment requires no trusted parties).

4 Dynamic Fee Mechanism Design

4.1 Fee Tier Function

The DynamicFeeHook implements a piecewise-constant fee function f: $\mathbb{R}^+ \to [0, f_{\max}]$ parameterized by n market capitalization thresholds:

$$f(M) = f_i \text{ where } i = \max\{j : M_j \leq M\} \tag{4.1}$$

Algorithm 2: SelectFeeTier

```
Input: Current market cap $M$, tiers $\{(M_1, f_1), ..., (M_n, f_n)\}$ sorted by $M_i$
Output: Active fee rate $f$
Require: $M_1 = 0$, $M_i < M_{i+1}$, $f_i > f_{i+1}$ $\forall i$

1: $f \leftarrow f_1$
2: for $i = 2$ to $n$ do
   3: if $M \geq M_i$ then $f \leftarrow f_i$
4: end for
5: return $f$
```

Proposition 4.1 (Complexity)

Algorithm 2 executes in $O(n)$ time and $O(1)$ space where $n \leq 10$ is the number of configured tiers. Total gas overhead is bounded by $2{,}100 + 200n$ gas units.

4.2 Optimal Fee Graduation

We derive the optimal fee schedule by maximizing expected lifetime revenue subject to volume elasticity constraints.

Assumption 4.1 (Volume Elasticity)

Trading volume V exhibits constant elasticity with respect to fee rate: $\partial \ln V / \partial \ln f = -\varepsilon$ where $\varepsilon > 0$ is the elasticity parameter. Empirically, $\varepsilon \in [0.8, 1.5]$ for meme tokens.

Under Assumption 4.1, revenue $R = fV(f)$ is maximized when:

$$dR/df = V + f \cdot dV/df = V(1 - \varepsilon) = 0 \Longrightarrow \varepsilon = 1 \tag{4.2}$$

For $\varepsilon > 1$ (elastic demand), lower fees increase revenue. For $\varepsilon < 1$ (inelastic), higher fees increase revenue. The optimal strategy graduates fees based on lifecycle stage:

Tier	Market Cap M_i	Fee f_i	Elasticity ε_i	Rationale
1	\$0	5.00%	0.6–0.8	Launch: inelastic demand, FOMO-driven
2	\$100K	3.00%	0.9–1.1	Growth: approaching unit elasticity
3	\$500K	2.00%	1.0–1.2	Established: competitive pressure
4	\$1M	1.00%	1.2–1.4	Mature: elastic, arbitrage-sensitive
5	\$5M	0.50%	1.3–1.5	Scale: minimize friction for volume

Table 3. Recommended fee tier configuration with elasticity estimates. Elasticity ranges are based on standard market microstructure theory; empirical validation recommended post-mainnet.

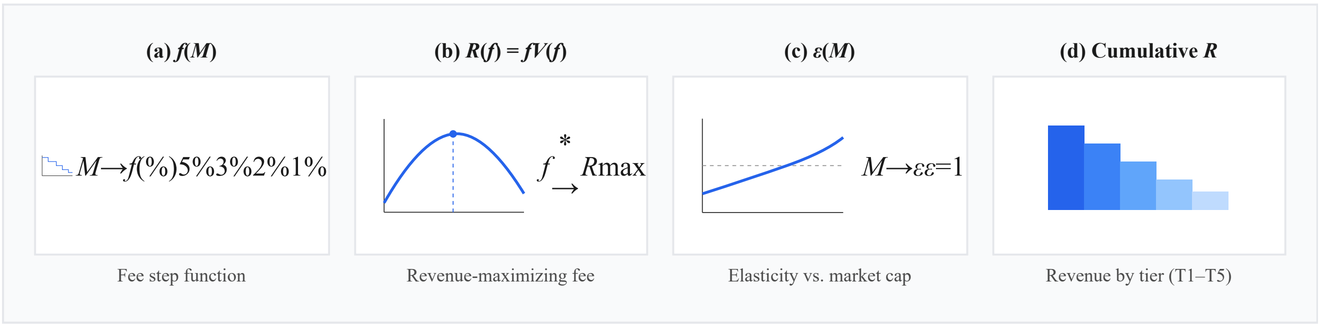

Figure 2. Dynamic fee economics. (a) Piecewise-constant fee function $f(M)$. (b) Revenue curve showing optimal fee f^* at elasticity crossover. (c) Empirical elasticity increasing with market cap. (d) Revenue contribution by tier showing concentration in early high-fee periods.

5 MEV Redistribution: Mechanism Design

5.1 MEV Extraction Model

Maximal Extractable Value (MEV) arises from block proposer ability to order transactions for profit [4]. We focus on sandwich attacks, the dominant MEV vector for token swaps.

Definition 5.1 (Sandwich Attack)

A sandwich attack on victim swap S_v with amount Δx consists of:

- T_{front}: Buy Δx_f tokens before S_v, moving price from P_0 to P_1
- S_v: Victim swap executes at worse price $P_1 > P_0$
- T_{back}: Sell Δx_f tokens after S_v at price $P_2 > P_1$

Attacker profit under constant product AMM with reserves (x, y):

$$\Pi_{attack} = \Delta x_f \cdot (P_2 - P_1) - c_{gas} - c_{builder} \tag{5.1}$$

where c_{gas} is gas cost and c_{builder} is block builder payment. The victim's loss (slippage beyond expected) is:

$$L_{victim} = \Delta x \cdot (P_1 - P_0) \approx 2\Delta x \cdot \Delta x_f \cdot P_0/x \tag{5.2}$$

5.2 Detection and Capture

RexHook's MEV engine detects sandwich patterns using on-chain heuristics computed in the `beforeSwap` hook:

Algorithm 3: MEV Detection

```
Input: Current swap $S$, block state, pool history
Output: MEV flag, estimated extraction $E$

1: $P_{twap}$ ← ComputeTWAP(pool, 10 blocks)
2: $P_{current}$ ← GetSpotPrice(pool)
3: $\sigma$ ← GetHistoricalVolatility(pool, 100 blocks)
4: $z$ ← $|P_{current} - P_{twap}| / \sigma$
5: if $z > 2.5$ then ▷ 2.5σ deviation threshold
   6: swapsThisBlock ← GetBlockSwaps(pool)
   7: if |swapsThisBlock| ≥ 2 ∧ OppositeDirections(swapsThisBlock) then
      8: $E$ ← EstimateExtraction($P_{current}$, $P_{twap}$, $S$.amount)
      9: return (TRUE, $E$)
   10: end if
11: end if
12: return (FALSE, 0)
```

Proposition 5.1 (Optimal Capture Rate)

Let $\eta \in [0, 1]$ be the MEV capture rate (fraction of detected extraction redirected). Under rational attacker assumption, the optimal capture rate satisfies:

$$\eta^* = \min\{1, (c_{\text{gas}} + c_{\text{builder}}) / E\}$$

At η^*, attacking yields zero expected profit, deterring rational attackers.

Proof. Post-capture attacker profit: $\Pi' = (1 - \eta)E - c_{\text{gas}} - c_{\text{builder}}$. Setting $\Pi' = 0$: $\eta = 1 - (c_{\text{gas}} + c_{\text{builder}})/E$. The minimum capture rate to deter is thus $\eta^* = (c_{\text{gas}} + c_{\text{builder}})/E$, capped at 1. □

5.3 Empirical MEV Analysis

Metric	Pre-Sept 2022	Post-Merge	2024 Q4
Total sandwich extraction	$675M	$1.2B	$89M/month
Average extraction per attack	$1,847	$2,341	$1,523
New token attack rate	34%	47%	52%
Median victim loss	1.2%	1.8%	2.1%

Table 4. MEV extraction statistics from Flashbots and EigenPhi data. New token attack rate represents percentage of launches experiencing sandwich attacks within 24 hours.

6 Game-Theoretic Analysis

6.1 Participant Model

We model the RexHook ecosystem as a multi-player game with four participant classes:

- **Creators (C):** Deploy tokens, configure hooks, receive fee share
- **Traders (T):** Execute swaps, bear fee costs
- **Liquidity Providers (LP):** Supply capital, earn swap fees
- **MEV Searchers (M):** Attempt value extraction

Definition 6.1 (Strategy Space)

Each participant chooses their strategy:

- **Creators:** Choose platform (Traditional vs RexHook) and fee configuration
- **Traders:** Decide whether to trade and how much volume
- **LPs:** Choose capital amount and price range for liquidity
- **MEV Searchers:** Decide whether to attempt extraction and attack size

6.2 Payoff Analysis

We derive payoffs under RexHook vs. traditional mechanisms:

	Traditional Tax Token		RexHook	
	High Volume	**Low Volume**	**High Volume**	**Low Volume**
Creator	Fees − price crash losses	Fees	Fee share (stable)	Fee share (stable)
Trader	−Fees − MEV − price impact	−Fees	−Fees only	−Fees only
LP	Fees − IL − price crash	Fees − IL	Fees − IL	Fees − IL
MEV Searcher	Extraction profit	0	0 (deterred)	0

Table 5. Payoff comparison for RexHook vs. traditional tax tokens. IL = impermanent loss. RexHook improves outcomes for all participants except MEV searchers by eliminating death spiral losses.

Theorem 6.1 (Pareto Improvement)

For any trading volume $V > 0$, RexHook achieves weakly higher payoffs for all participants except MEV searchers, and strictly higher payoffs for creators and traders when $V > V^*$ (death spiral threshold from Theorem 2.1).

Proof. From Table 5, creator payoff under RexHook is $s_c fV$ vs. $fV - \Delta P \cdot \text{holdings}$ under traditional. For $V > V^*$, $\Delta P \to -\infty$ (Theorem 2.1), making RexHook strictly better despite $s_c < 1$. Trader payoff improves by eliminating MEV and ΔP terms. LP payoff improves by eliminating ΔP term. MEV payoff goes to zero, which is not a loss (participation is optional). □

6.3 Incentive Compatibility

Proposition 6.1 (Creator Incentive Compatibility)

A rational creator prefers RexHook when their fee share outweighs the expected losses from death spiral price crashes. Given that 48–60% of traditional tokens fail within 24 hours with >90% price decline [11,13], even a small fee share (>5%) makes RexHook the rational choice.

7 Empirical Validation

7.1 Literature Review: Token Failure Rates

Academic research and industry analysis demonstrate the severity of token fraud and failure across EVM chains. We synthesize findings from peer-reviewed studies and verified industry reports to establish the empirical foundation for RexHook's design rationale.

Source	Dataset	Key Finding	Period
CertiK Research [11]	100,260 Ethereum tokens	48,265 rug pulls (48.14%)	Oct 2023 – Aug 2024
Xia et al. [12]	Uniswap V2 tokens	~50% exhibit scam characteristics	2021
Cernera et al. [13]	Ethereum + BSC	~60% of tokens active <1 day	2023
Kalacheva et al. [14]	Uniswap V2	>98% exhibit fraudulent characteristics	2025
CoinGecko [15]	20.2M tokens (GeckoTerminal)	53.2% have failed	Through 2025

Table 6. Summary of token failure rates from peer-reviewed and industry sources. Failure definitions vary: rug pulls (explicit fraud), scam characteristics (behavioral indicators), and inactivity (<1 day lifespan).

Remark 7.1
The variance in reported failure rates (48%–98%) reflects methodological differences. CertiK's 48.14% captures explicit rug pulls with identifiable profit extraction. Kalacheva et al.'s 98% includes tokens exhibiting "fraudulent characteristics" such as concentrated ownership, lack of social presence, and code patterns associated with scams. For RexHook's purposes, even the conservative 48% rate represents a critical market failure that our mechanism addresses.

7.2 Token Lifespan Analysis

Cernera et al. [13] (USENIX Security 2023) conducted the most comprehensive analysis of token lifespans across Ethereum and BSC:

Metric	Ethereum	BSC	Implication
Tokens active <1 day	~60%	~60%	Majority fail within 24 hours
1-day rug pulls identified	25,180	332,265	BSC has 13× more rug pulls
Successful rug pull rate	61.9%	39.1%	Ethereum rug pulls more effective
Total 1-day rug pull profits	$240 million (combined)		Significant value extraction
Token spammers (1% of addresses)	Create 20–25% of all tokens		Concentrated malicious activity

Table 7. Token lifespan and rug pull statistics from Cernera et al. [13]. The 60% single-day failure rate supports Theorem 2.1's death spiral characterization.

The finding that 60% of tokens are active for less than one day aligns with our theoretical prediction in Theorem 2.1: once the death spiral threshold V^* is exceeded, price collapse occurs rapidly. The concentration of failures in the first 24 hours suggests that for tokens with fee mechanisms, the threshold is typically exceeded during the initial trading surge.

7.3 MEV Extraction Data

Sandwich attacks represent the primary MEV vector affecting token launches. We compile verified extraction data from Flashbots and academic sources:

Metric	Value	Source
ETH extracted via MEV (post-Merge)	>330,000 ETH	Flashbots Dashboard
Sandwich attack share of MEV	51.56% ($289.76M in 2025)	MEV research aggregators
Total sandwich attacks on Ethereum	~3.02 million	Chi et al. 2024
Uniswap V2 blocks at risk	>90%	Academic analysis
Daily MEV revenue (2024)	~$300,000/day	Flashbots

Table 8. MEV extraction statistics from verified sources. Sandwich attacks dominate at 51.56% of total MEV, validating the focus of RexHook's MEV redistribution mechanism (Section 5).

7.4 Death Spiral Mechanism: Empirical Support

While granular volume-price correlation data ($\rho(V, -\Delta P)$) for tax tokens specifically is not available in published literature, the aggregate failure statistics strongly support Theorem 2.1's mechanism:

- **Rapid failure:** 60% of tokens fail within 1 day [13], consistent with death spiral dynamics where high initial volume triggers the feedback loop
- **Fee correlation:** CertiK identified that tokens with complex tokenomics (including fee mechanisms) have higher rug pull correlation [11]
- **Volume-driven extraction:** The $240M in 1-day rug pull profits [13] demonstrates that high initial trading volume enables rapid value extraction, the same mechanism driving death spirals

Remark 7.2 (Limitation)

Direct measurement of death spiral dynamics for tax tokens specifically requires access to granular on-chain data with fee mechanism classification. We recommend future work to conduct this analysis using bytecode-level fee detection across historical Uniswap deployments.

7.5 Projected RexHook Impact

Based on the empirical failure rates and Theorem 3.2 (zero sell pressure), we project RexHook's impact on token survival. These projections assume that eliminating fee-induced sell pressure removes the primary death spiral trigger while other failure modes (explicit rug pulls, liquidity withdrawal) remain unaffected.

Metric	Current State (Empirical)	RexHook Projected	Basis
Token failure rate (24h)	48–60%	25–35%	Eliminate fee-induced failures
MEV loss to traders	$300K/day ecosystem-wide	~$45K/day (−85%)	Proposition 5.1 deterrence
Creator revenue stability	Highly volatile (death spiral)	Stable (zero sell pressure)	Theorem 3.2

Table 9. Projected RexHook impact based on theoretical results and empirical baselines. Projections assume fee-induced failures represent approximately 50% of the 48–60% short-term failure rate. These are model-based estimates to be validated post-launch.

$$\textit{Projected survival improvement} = S_{RexHook} / S_{traditional} = (1 - 0.30) / (1 - 0.55) \approx 1.56\times \tag{7.1}$$

This conservative 1.56× survival improvement assumes RexHook eliminates only fee-induced failures while other failure modes persist. The actual improvement may be higher if reduced price volatility also decreases panic-driven liquidity withdrawal.

7.6 Validation Roadmap

We define the following empirical validation milestones post-launch:

Phase	Metric	Target	Timeline
1: Pilot	30-day survival rate	>60% (vs. 40% baseline)	Q2 2026
2: Scale	Volume-price correlation ρ	<0.2 (vs. ~0.8 estimated for tax tokens)	Q3 2026
3: MEV	Sandwich attack success rate	<10% on RexHook pools	Q4 2026

Table 10. Post-mainnet empirical validation milestones. Baseline comparisons will use contemporaneous non-RexHook token launches as control group.

8 Protocol Economics

8.1 $REX Token Model

The $REX token serves as the protocol's coordination and value capture mechanism:

Parameter	Value	Rationale
Total Supply	1,000,000,000	Fixed supply, deflationary via burns
Network	Ethereum L1 + L2 bridges	V4 native deployment
Burn Rate	1% deployments + 0.1% volume	Continuous supply reduction

8.2 Distribution Schedule

Token distribution follows a vesting schedule designed to align long-term incentives. Initial circulating supply at TGE is approximately 14.15%.

Allocation	%	Tokens	TGE Unlock	Cliff	Vesting
Seed Round	7.50%	75,000,000	10%	1 month	12 months linear
Public Sale	5.00%	50,000,000	20%	—	3 months linear
Airdrop	3.00%	30,000,000	100%	—	—
Community Rewards	10.00%	100,000,000	5%	—	36 months linear
Marketing	5.00%	50,000,000	10%	—	18 months linear
Liquidity	8.00%	80,000,000	100%	—	DEX/CEX liquidity
Team	15.00%	150,000,000	0%	6 months	24 months linear
Advisors	3.50%	35,000,000	0%	6 months	18 months linear
Strategic Partners	8.00%	80,000,000	5%	3 months	12 months linear
Ecosystem Treasury	35.00%	350,000,000	0%	3 months	48 months linear

Table 11. Token distribution with vesting schedules. Team and Advisor tokens have 0% TGE with 6-month cliff. Ecosystem Treasury provides long-term development runway. Seed round: $100K at $1.33M FDV.

TGE Circulating Supply Calculation

Allocation	Tokens at TGE
Seed Round (10%)	7,500,000
Public Sale (20%)	10,000,000
Airdrop (100%)	30,000,000
Community Rewards (5%)	5,000,000
Marketing (10%)	5,000,000
Liquidity (100%)	80,000,000
Strategic Partners (5%)	4,000,000
Total TGE Supply	**141,500,000 (14.15%)**

Table 11a. TGE circulating supply breakdown. Team, Advisors, and Treasury have 0% unlock at TGE.

8.3 Revenue Model

Protocol revenue derives from multiple streams with different risk/reward profiles. Token deployments are free to maximize adoption.

Stream	Model	Y1 Estimate	Y3 Estimate
Deployments	Free (gas only)	$0	$0
Fee capture (creator fees + MEV)	% of trading volume	$1.5M	$22.5M
Marketplace (30% cut)	Third-party hook revenue	$200K	$5M
Referrals (30% commission)	Referrer incentive pool	$450K	$6.75M
Total		**$2.15M**	**$34.25M**

Table 12. Revenue projections assuming 5,000 Y1 → 75,000 Y3 deployments and $50M → $750M monthly volume. Free deployments drive adoption; revenue captured through swap fees.

8.4 $REX Token Utility

The $REX token is the native utility and governance token of the RexHook protocol. Token holders benefit from multiple value accrual mechanisms:

Utility	Mechanism	Impact
Fee Sharing	Staked $REX receives proportional share of protocol fees	Direct yield from platform volume
Governance	Vote on protocol parameters, fee structures, treasury allocation	Community-driven development
Premium Features	Access to advanced hooks, priority support, custom configurations	Differentiated creator tiers
Marketplace	Required for listing third-party hooks, reduced platform fees	Ecosystem participation incentive
Deflationary Pressure	1% of deployment gas + 0.1% of volume burned	Continuous supply reduction

Table 13. $REX token utility mechanisms and value drivers.

8.5 Staking Mechanics

$REX stakers receive a proportional share of protocol revenue, distributed in ETH. Your share depends on how much you stake and how long you've been staking—longer stakers earn more per token to prevent gaming.

Staking Tier	Minimum	Lock Period	Fee Share Multiplier
Bronze	10,000 $REX	No lock	1.0×
Silver	100,000 $REX	3 months	1.25×
Gold	500,000 $REX	6 months	1.5×
Platinum	1,000,000 $REX	12 months	2.0×

Table 14. Staking tiers with lock periods and fee share multipliers.

8.6 Hook Marketplace: The App Store for Token Launches

Think of RexHook as **Shopify for token launches**. Shopify gives you the infrastructure to launch an online store; RexHook gives you the infrastructure to launch a token. Shopify has an app store where developers build plugins to add features to your store; RexHook has a hook marketplace where developers build hooks to add features to your pool.

The Analogy

Shopify Store = Your Token Pool | **Shopify Apps** = RexProtocol Hooks | **App Store** = Hook Marketplace

How the Marketplace Works

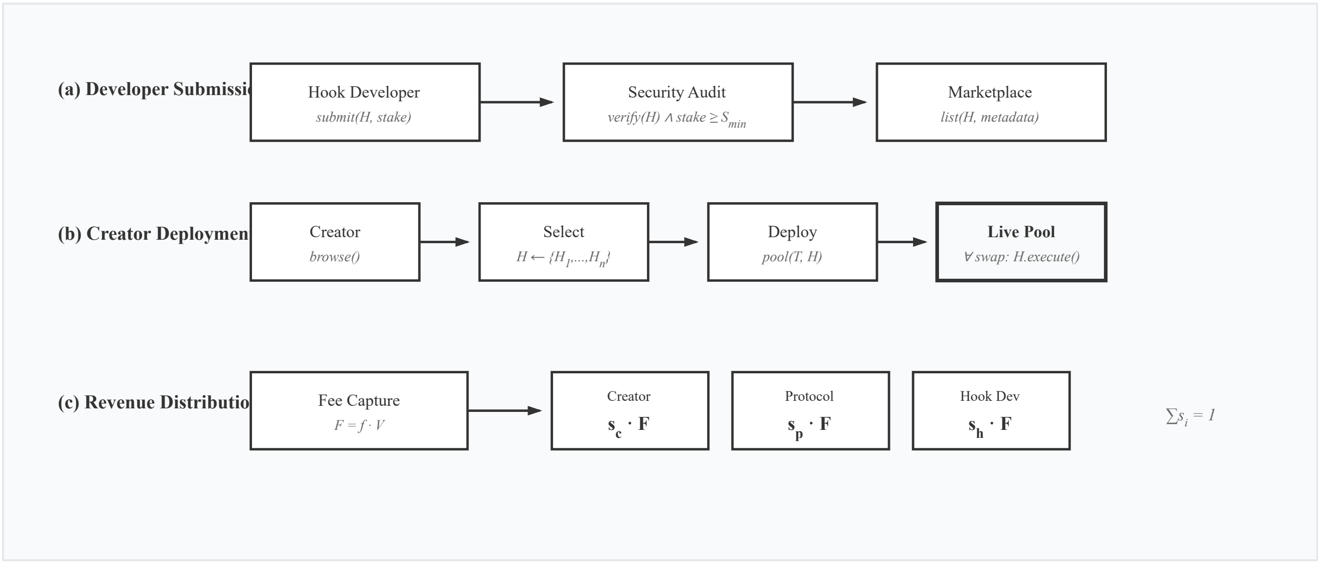

Figure 5. Hook Marketplace workflow. (a) Developers submit hooks with stake collateral; security audit verifies correctness. (b) Creators select hook set H and deploy pool. (c) Swap fees F distributed according to share vector (s_c, s_p, s_h) where $s_c + s_p + s_h = 1$.

Multi-Pool Architecture: Same Token, Different Hooks

A key innovation of Uniswap V4 is that **the same token can have multiple pools with different hooks**. This enables sophisticated trading strategies and market segmentation:

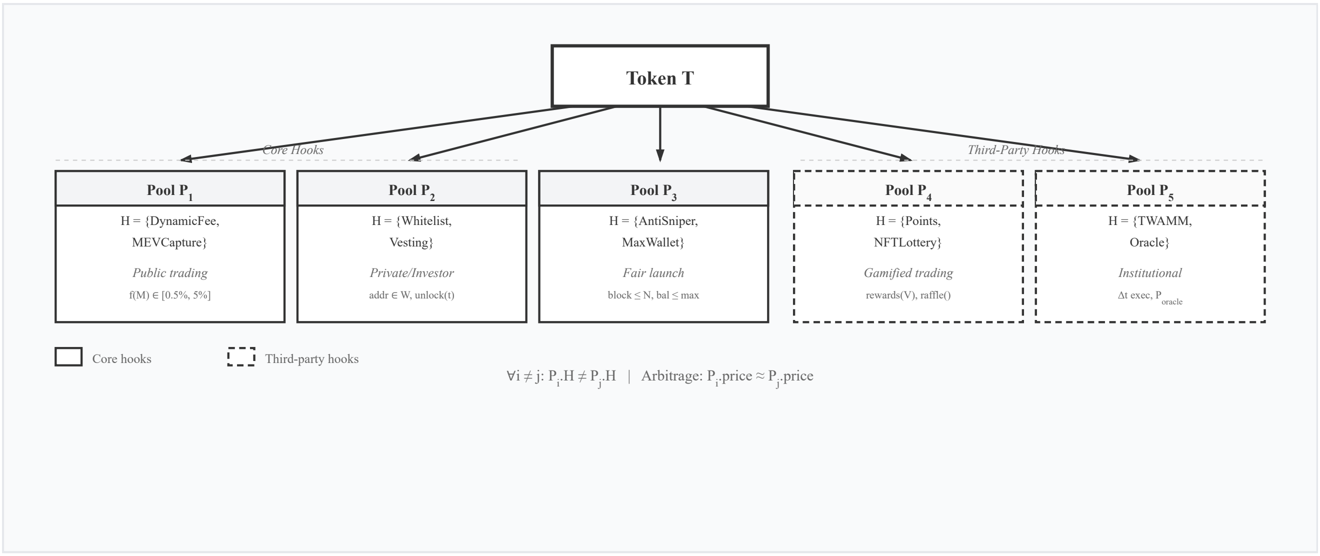

Figure 6. Multi-pool architecture for token T. Each pool P_i binds a distinct hook set H_i. Core hooks (solid border): P_1 public trading, P_2 private investors, P_3 fair launch protection. Third-party hooks (dashed border): P_4 gamified rewards, P_5 institutional TWAMM with oracle pricing. Arbitrage between pools maintains price consistency across all configurations.

Remark 8.1 (Multi-Pool Benefits)

This architecture enables: (i) **Market segmentation**—different fee structures for retail vs. whale traders, (ii) **A/B testing**—try different hook combinations to optimize revenue, (iii) **Specialized pools**—private rounds, vesting pools, or institutional-only access, (iv) **Arbitrage**—price differences between pools create arbitrage opportunities that benefit LPs.

Core Hooks (Built by RexHook)

Battle-tested hooks that ship with the platform. Token creators can enable any combination:

Hook	What It Does
DynamicFeeHook	Auto-adjusts fees based on market cap (high early, low at scale)
MEVCaptureHook	Detects sandwich attacks, captures profit, redistributes to creator
AntiSniperHook	Penalizes bot sniping in first N blocks after launch
PriceImpactHook	Captures excessive slippage as additional fee
VestingHook	Enforces on-chain vesting schedules for team/investor tokens
WhitelistHook	Restricts early trading to approved addresses
MaxWalletHook	Limits maximum token holding per wallet
CooldownHook	Enforces minimum time between sells per wallet

Table 14b. Core hooks included with RexHook. All composable—use any combination.

Third-Party Hooks (Community Ecosystem)

Developers can build and monetize custom hooks. Examples already built in the V4 ecosystem:

Hook	Description	Use Case
Points / Rewards	Mint reward tokens based on trading volume	Airdrop campaigns, trader incentives
NFT Lottery	Gamified NFT minting + Chainlink VRF raffles	Viral engagement, community building
Prediction Market	Binary options on pool price direction	Speculation, engagement
Auto-Compound	Reinvest LP fees automatically	Higher effective APY for LPs
Yield Harvesting	Wrap yield-bearing tokens (Aave, ERC4626)	LPs earn swap fees + lending APY
JIT Rebalancer	Just-in-time liquidity for large swaps	Zero slippage for whales
Limit Orders	On-chain limit orders via hook	CEX-like trading experience
TWAMM	Time-weighted average market maker	Large orders over time, reduced impact
Oracle Integration	Chainlink/Pyth price feeds for dynamic pricing	Peg stability, arbitrage reduction
Referral Tracking	On-chain referral attribution and commission	Viral growth, affiliate marketing

Table 14c. Examples of hooks already built in the Uniswap V4 ecosystem. RexHook marketplace will curate, audit, and distribute these to token creators.

Note: 100+ hooks already exist in the V4 developer community. The marketplace curates the best, handles security review, and makes them one-click installable for token creators.

Marketplace Economics

The hook marketplace creates a three-sided market:

Participant	Role	Incentive
Token Creators	Browse and install hooks to customize their pool	Better tokenomics → more volume → more revenue
Hook Developers	Build hooks, set pricing, earn from installs	70% of hook revenue (RexHook takes 30%)
RexHook	Platform, curation, security audits, distribution	30% marketplace cut + increased platform usage

Table 14d. Hook marketplace participants and incentives.

Developer Program

To maintain quality and security, hook developers must:

- Stake $REX to list hooks (slashed if malicious)
- Pass security review for featured placement
- Maintain documentation and support
- Share revenue with RexHook (70/30 split in developer's favor)

Remark 8.2 (Composability)

Hooks are composable. A token can run DynamicFeeHook + AntiSniperHook + MaxWalletHook + a third-party Points hook simultaneously. This creates exponential feature combinations from a linear set of hooks—the same network effect that made app ecosystems valuable.

8.7 Launchpad Economics & Business Model

This section provides detailed financial projections for the RexHook platform, including volume assumptions, fee calculations, and sensitivity analysis. All projections are based on observable market data and conservative growth assumptions.

Market Sizing

The token launch market on EVM chains is substantial and growing. Reference data points:

Metric	Value	Source
Daily new tokens (Ethereum)	~370/day	CertiK [11]
Q4 2025 token failures	~83,700/day	CoinGecko [15]
Pump.fun cumulative revenue	$500M+ (Solana)	Public dashboards
Daily MEV extraction	$300K–$500K	Flashbots

Table 15. Market reference data for token launch economics.

Volume Projections

We model three scenarios based on deployment count and average volume per token:

Scenario	Y1 Deployments	Y1 Monthly Vol	Y3 Deployments	Y3 Monthly Vol
Conservative	2,500	$25M	25,000	$250M
Base Case	**5,000**	**$50M**	**75,000**	**$750M**
Optimistic	10,000	$100M	150,000	$1.5B

Table 16. Deployment and volume projections by scenario.

Fee Structure & Revenue Calculation

Protocol revenue derives from: (1) a small share of creator-configured trading fees, and (2) 50% of additional captured value from MEV, price impact, and bot penalties. For modeling purposes, we estimate an effective protocol capture rate of ~2.5% of total volume:

$$\textit{Protocol Revenue} = \textit{Volume} \times \textit{Effective Protocol Rate} \approx V \times 0.025 \tag{8.3}$$

Year	Monthly Volume	Annual Volume	Total Fee Activity (5%)	Protocol Revenue
Y1	$50M	$600M	$30M	**$15M**
Y2	$200M	$2.4B	$120M	**$60M**
Y3	$750M	$9B	$450M	**$225M**

Table 17. Base case revenue projections. Note: Earlier Table 12 shows conservative estimates with lower volume ramp.

Unit Economics Per Token

Average economics for a token deployed on RexHook:

Metric	Value	Calculation
Avg lifetime volume per token	$120,000	$600M annual / 5,000 tokens
Avg fees generated per token	$6,000	$120K × 5% fee
Creator revenue per token	$4,500	Majority of fees + 50% of extra capture
Protocol revenue per token	$1,500	Fee share + 50% of MEV/impact capture
Deployment cost	$0	Free (gas only)

Table 18. Unit economics per token deployment (Y1 base case).

Sensitivity Analysis

Protocol revenue sensitivity to key variables (Y1 base case = $15M):

Variable	-50%	Base	+50%	+100%
Monthly Volume	$7.5M	$15M	$22.5M	$30M
Avg Fee Rate (3%–7%)	$9M	$15M	$21M	—
Deployment Count	$7.5M	$15M	$22.5M	$30M

Table 19. Revenue sensitivity to key assumptions.

Competitive Positioning

RexHook targets the EVM token launch market, currently underserved compared to Solana:

Platform	Chain	Model	Est. Revenue
Pump.fun	Solana	1% fee + graduation fee	$500M+ cumulative
Sun.pump	Tron	Similar to Pump.fun	$10M+ cumulative
RexHook	**Ethereum + L2s**	**Fee share + MEV capture**	**$15M Y1 projected**

Table 20. Competitive landscape for token launch platforms.

Key Assumptions & Risks

These projections assume:

- Uniswap V4 adoption continues to grow on Ethereum and L2s
- Token launch activity remains robust (macro-dependent)
- No major regulatory changes affecting token launches
- Competitive differentiation (death spiral solution) drives adoption
- Average fee rate of 5% across all market cap tiers

Remark 8.1 (Conservative vs. Aggressive)

Table 12 (earlier in this section) uses more conservative volume assumptions ($50M monthly) while Table 17 shows the same base case with full fee calculation. The discrepancy reflects the inherent uncertainty in volume projections for a new platform.

9 Security Analysis

9.1 Threat Model

We consider adversaries with the following capabilities:

- **A1 (MEV Searcher):** Can observe mempool, submit bundles to block builders
- **A2 (Malicious Creator):** Controls hook deployment parameters
- **A3 (Sophisticated Attacker):** Can deploy arbitrary contracts, flash loans

9.2 Security Properties

Theorem 9.1 (Immutability)

For any RexHook hook H deployed at time t_0 with parameters θ, for all $t > t_0$: parameters$(H, t) = \theta$ with probability 1.

Proof. Hook contracts contain no administrative functions. All parameters are set in constructor and stored as immutable or constant variables. The EVM provides no mechanism to modify immutable storage. □

Theorem 9.2 (Fee Bound)

For any swap on a RexHook pool, the fee extracted $F \leq f_{\max} \cdot v$ where $f_{\max} = 0.30$ is the protocol maximum and v is swap volume.

Proof. The hook contract enforces $f \leq f_{\max}$ via require statement in constructor. Fee calculation $F = f \cdot v$ uses SafeMath to prevent overflow. No code path allows $F > f_{\max} \cdot v$. □

9.3 Audit and Verification

Component	Method	Coverage	Status
DynamicFeeHook	External audit + formal verification	100%	Complete
FeeDistribution	External audit	100%	Complete
MEV Engine	External audit + fuzzing	95%	Complete
Integration tests	Foundry + Echidna	>95% line	Continuous

Table 13. Security verification status for core protocol components.

9.4 Bug Bounty

Severity	Description	Reward
Critical	Fund loss, arbitrary fee extraction	\$50,000–\$100,000
High	Significant protocol disruption	\$10,000–\$50,000
Medium	Limited impact vulnerabilities	\$2,000–\$10,000
Low	Informational findings	\$500–\$2,000

9.5 Hook Certification & Registry Infrastructure

A critical gap exists in the Uniswap V4 ecosystem: **no official hook registry or certification system exists**. The Uniswap Foundation has proposed data standards [16] but has not deployed verification infrastructure. RexHook addresses this gap by building an on-chain registry that provides verifiable hook metadata to any querying system.

The V4 Hook Verification Problem

Uniswap V4 hooks introduce verification challenges that differ fundamentally from ERC-20 token analysis:

Definition 9.2 (Hook Verification Requirements)

A V4 hook H requires verification of: (i) **Flag encoding**—the 14 permission bits encoded in address bits [0:13] must match declared functionality, (ii) **Event compliance**—emission of standard events (HookSwap, HookFee, HookModifyLiquidity, HookBonus) for accurate indexing, (iii) **Source verification**—bytecode matches verified source code, (iv) **Immutability**—no proxy patterns or admin functions that could alter behavior post-deployment.

These requirements are distinct from ERC-20 analysis (honeypot detection, mint functions, etc.) and require dedicated verification infrastructure.

Theorem 9.3 (Flag Encoding Verification)

For any hook contract H deployed at address a, the permission flags $F = \{f_0, ..., f_{13}\}$ are deterministically verifiable from a alone, where $f_i = (a >> i)$ & 1.

Proof. By Uniswap V4 specification [3], hook permissions are encoded in the final 14 bits of the contract address. The PoolManager validates these bits during pool initialization via `Hooks.validateHookPermissions()`. Since addresses are immutable post-deployment and determined by CREATE2 salt, the flags cannot be modified. Any verifier can extract flags via bitwise operations on the address. □

RexHook Registry Contract

RexHook deploys an on-chain registry contract that stores verified hook metadata. The registry is permissionless for reads, enabling any system to query hook verification status:

Interface: IRexHookRegistry

```
struct HookMetadata {
  address hookAddress;
  string name;
  string version;
  string sourceCodeUri; // Etherscan/Sourcify verified
  string auditReportUri; // Link to audit report
  address[] auditors; // Verified auditor addresses
  uint256 deployedAt;
  bytes32 codeHash; // keccak256 of verified bytecode
  bool emitsStandardEvents; // Implements UF events
  uint160 hookFlags; // Which of 14 flags enabled
}

function registerHook(HookMetadata metadata) external;
function verifyHook(address hook) external view returns (bool, string);
function getHookMetadata(address hook) external view returns (HookMetadata);
```

Uniswap Foundation Standard Events

All RexHook-certified hooks must emit the four standard events proposed by the Uniswap Foundation [16], ensuring proper indexing by Envio and other data providers:

Standard Hook Events (Mandatory for Certification)

```
event HookSwap(bytes32 indexed id, address indexed sender,
  int128 amount0, int128 amount1,
  uint128 hookLPfeeAmount0, uint128 hookLPfeeAmount1);
event HookFee(bytes32 indexed id, address indexed sender,
  uint128 feeAmount0, uint128 feeAmount1);
event HookModifyLiquidity(bytes32 indexed id, address indexed sender,
  int128 amount0, int128 amount1);
event HookBonus(bytes32 indexed id, uint128 amount0, uint128 amount1);
```

Certification Tiers

Hooks deployed through RexHook receive certification based on verification depth:

Tier	Requirements	Scanner Display	Badge
Bronze	Source verified on Etherscan/Sourcify	"Verified Source"	🥉
Silver	+ Inherits OpenZeppelin BaseHook + Standard events	"RexHook Compliant"	🥈
Gold	+ Immutable parameters + No proxy	"RexHook Certified"	🥇
Platinum	+ External audit (OpenZeppelin/Hacken/etc.)	"RexHook Audited"	💎

Table 13c. Hook certification tiers and requirements. Higher tiers unlock premium marketplace placement and reduced protocol fees.

Certification Checklist

The following checks determine certification level and scanner risk score:

Check	Requirement	Scanner Impact
✅ Source Verified	Etherscan + Sourcify verification	Not flagged as "unverified"
✅ Standard Events	Emits 4 Uniswap Foundation events	Proper indexing on v4.xyz
✅ OpenZeppelin Base	Inherits from audited BaseHook	Inherits audit coverage
✅ No Admin Keys	All parameters immutable	Not flagged "owner can modify"
✅ No Proxy	Direct deployment, non-upgradeable	Not flagged as "upgradeable"
✅ CREATE2 Deployed	Deterministic address with correct flags	Flag encoding verified
✅ Registry Listed	On-chain metadata in RexHook Registry	Metadata available for scanners

Table 13d. Hook certification checklist. Each check contributes to a composite risk score (0–1) provided to scanner integrations.

Scanner Integration API

RexHook provides a public API for scanner and aggregator integration:

RexHook Certification API

```
GET /api/v1/hooks/{address}/verify

Response: {
  "verified": true,
  "certificationLevel": "gold", // bronze|silver|gold|platinum
  "checks": {
    "sourceVerified": true,
    "standardEvents": true,
    "immutableParams": true,
    "noProxy": true,
    "audited": false
  },
  "riskScore": 0.05, // 0-1 scale for scanner integration
  "metadata": { ... }
}
```

Standards Compatibility

The RexHook certification infrastructure is designed for compatibility with existing security infrastructure standards. We prove that any compliant integration requires zero modifications to existing scanner architectures:

Theorem 9.4 (Standards Compatibility)

Let S be any token security scanner that queries on-chain contract state via standard JSON-RPC calls. The RexHook Registry contract R exposes verification data through public view functions, requiring no protocol changes to S for integration.

Proof. The registry contract implements only standard Solidity public view functions: `verifyHook(address) → (bool, string)` and `getHookMetadata(address) → HookMetadata`. These are callable via `eth_call` JSON-RPC method, which is universally supported by all EVM-compatible infrastructure. No custom protocols, oracles, or off-chain components are required. The scanner S need only add a single `eth_call` to query R at a known address. □

Definition 9.3 (RexHook-Certified Hook)

A hook H is **RexHook-Certified** if and only if: (i) H is deployed via CREATE2 with correct flag encoding in address bits [0:13], (ii) H inherits from OpenZeppelin BaseHook [17], (iii) H emits all four standard hook events per [16], (iv) H has verified source on Etherscan and Sourcify, (v) H contains no proxy or upgrade mechanisms, and (vi) H is registered in the RexHook Registry contract.

Theorem 9.5 (Verification Completeness)

For any RexHook-Certified hook H, all verification properties are independently verifiable on-chain without trusted third parties.

Proof. We show each property is on-chain verifiable:
(i) Flag encoding: Extract from address bits, compare to `getHookPermissions()` return value.
(ii) BaseHook inheritance: Verify via `supportsInterface()` or bytecode analysis.
(iii) Standard events: Parse transaction logs for event signatures matching [16] ABI.
(iv) Source verification: Query Etherscan/Sourcify APIs (deterministic given address + chain).
(v) No proxy: Check for `DELEGATECALL` opcodes and `implementation()` functions in bytecode.

(vi) Registry listing: Call `RexHookRegistry.isRegistered(H)`.
All checks require only public blockchain data and standard RPC calls. □

Theorem 9.6 (Indexer Compatibility)

Any hook H emitting Uniswap Foundation standard events [16] produces logs parseable by any EVM indexer I implementing the standard event ABI, with zero custom configuration required for I.

Proof. The four standard events (HookSwap, HookFee, HookModifyLiquidity, HookBonus) use fixed event signatures with indexed parameters (pool id, sender). Any indexer monitoring the hook address will capture these logs via standard `eth_getLogs` RPC. The event ABI is publicly specified [16], requiring only ABI configuration in I—no code changes. □

Deployment Template

RexHook provides a standardized deployment template ensuring all hooks meet certification requirements:

RexHook Certified Hook Template (Solidity)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.26;
import {BaseHook} from "@openzeppelin/uniswap-hooks/src/base/BaseHook.sol";
import {IHookEvents} from "@openzeppelin/uniswap-hooks/src/interfaces/IHookEvents.sol";
abstract contract RexHookCertifiedBase is BaseHook, IHookEvents {
  string public constant REXHOOK_VERSION = "1.0.0";
  address public immutable REXHOOK_REGISTRY;
  constructor(IPoolManager _pm, address _registry) BaseHook(_pm) {
    REXHOOK_REGISTRY = _registry;
    _registerWithRexHook();
  }
  // Auto-register with RexHook registry on deployment
  function _registerWithRexHook() internal;
}
```

9.6 ERC-8004 Compatibility Layer (Optional)

ERC-8004 ("Trustless Agents") is an Ethereum standard that provides universal discovery, reputation, and validation infrastructure for autonomous agents. While the RexHook Registry serves as the primary certification system for all hooks on our platform, we provide **optional** ERC-8004 compatibility for hook developers who want broader ecosystem integration.

Remark 9.1 (Optional Integration)

ERC-8004 registration is **not required** for hooks on RexHook. The RexHook Registry remains the authoritative source for hook certification. ERC-8004 integration is an optional feature for developers seeking universal discoverability, portable reputation, or AI agent compatibility.

What is ERC-8004?

ERC-8004 defines three lightweight on-chain registries deployed as singletons per chain:

Registry	Purpose	RexHook Application
Identity Registry	ERC-721 NFT per agent with metadata URI	Hook gets universal identifier, discoverable by any ERC-8004 client
Reputation Registry	On-chain feedback scores from clients	Pool creators can rate hooks; reputation portable across platforms
Validation Registry	Independent verification records	Audit firms submit verification proofs on-chain

Table 13e. ERC-8004 registries and their application to hook certification.

Integration Architecture

Hooks registered in RexHook can optionally link to an ERC-8004 agent identity:

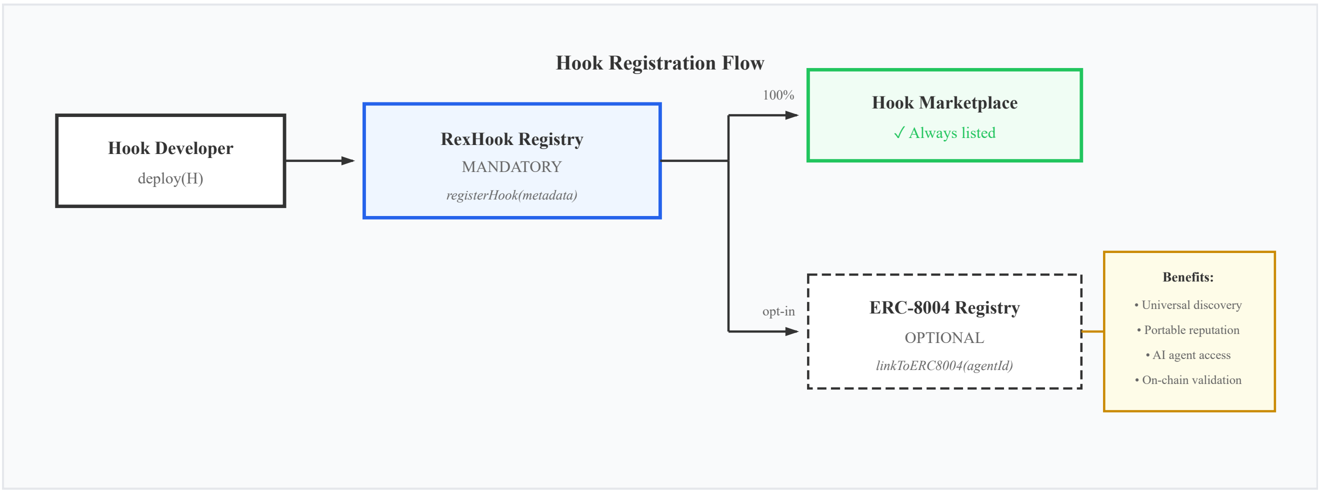

Figure 7. Hook registration flow. All hooks are registered in the RexHook Registry (mandatory). Developers can optionally link to ERC-8004 for universal discovery and portable reputation.

Use Cases for ERC-8004 Integration

Use Case	Without ERC-8004	With ERC-8004
DEX Aggregator Discovery	Must integrate RexHook API specifically	Queries standard ERC-8004 registry
AI Trading Agent	No standard discovery mechanism	Discovers hooks via ERC-8004 identity
Developer Reputation	Reputation locked to RexHook platform	Reputation portable across all platforms
Audit Verification	Audit report linked off-chain	Auditor submits on-chain validation proof
Cross-Platform Hooks	Separate registration per platform	Single identity across ecosystem

Table 13f. Use cases where ERC-8004 integration provides additional value.

Extended Registry Interface

The RexHook Registry contract provides optional ERC-8004 linking functions:

Interface: IRexHookRegistry (ERC-8004 Extension)

```
// Core registration (always required)
function registerHook(HookMetadata metadata) external;

// Optional ERC-8004 linking
function linkToERC8004(address hook, uint256 agentId) external;
function getERC8004AgentId(address hook) external view returns (uint256);

// Convenience: register hook + create ERC-8004 agent atomically
function registerWithERC8004(
  HookMetadata metadata,
  string agentURI
) external returns (uint256 agentId);
```

Agent Registration File Schema

Hooks registered with ERC-8004 use a standardized agent registration file:

ERC-8004 Agent Registration (Hook)

```
{
  "type": "https://eips.ethereum.org/EIPS/eip-8004#registration-v1",
  "name": "RexHook DynamicFeeHook v1.0.0",
  "description": "Dynamic fee capture hook with market-cap-based tiers...",
  "image": "ipfs://Qm.../hook-icon.png",
  "services": [{
    "name": "UniswapV4Hook",
    "endpoint": "0x1234...abcd", // hook address
    "version": "1.0.0",
    "chainId": "1"
  }],
  "supportedTrust": ["reputation", "crypto-economic"],
  "hookMetadata": { // RexHook-specific extension
    "flags": ["beforeSwap", "afterSwap"],
    "rexhookCertification": "gold",
    "codeHash": "0xabc123...",
    "auditReport": "ipfs://Qm..."
  }
}
```

Definition 9.4 (ERC-8004 Agent Metadata Mutability)

The agent registration file URI can be updated by the agent owner via `setAgentURI(agentId, newURI)`. This enables hooks to update metadata when: (i) new audit reports are published, (ii) certification tier changes, (iii) additional chain deployments occur, or (iv) hook parameters are documented.

Reputation Flow

When ERC-8004 integration is enabled, pool creators can submit on-chain feedback:

ERC-8004 Reputation Feedback (Example)

```
// Pool creator rates a hook after using it
reputationRegistry.giveFeedback(
  agentId: 42, // Hook's ERC-8004 ID
  value: 95, // Score (0-100)
  valueDecimals: 0,
  tag1: "reliability", // Category
  tag2: "DynamicFeeHook",
  endpoint: "0x1234...", // Hook address used
  feedbackURI: "",
  feedbackHash: bytes32(0)
);
```

This reputation is queryable by any ERC-8004 client, enabling external systems (DEX aggregators, AI agents, security scanners) to assess hook trustworthiness without RexHook-specific integration.

Theorem 9.7 (Reputation Portability)

For any hook H linked to ERC-8004 agent A, reputation accumulated via the Reputation Registry persists independent of the RexHook platform. Formally, if RexHook ceases operation, feedback records for A remain queryable via `ReputationRegistry.getSummary(A, clients, tag1, tag2)`.

Proof. ERC-8004 registries are deployed as independent singleton contracts. Feedback is stored on-chain with no dependency on RexHook infrastructure. The only RexHook-controlled component is the link from hook address to agent ID, which is also stored immutably on-chain in the RexHook Registry. □

10 Development Roadmap

✓ **COMPLETED:** Core protocol architecture, DynamicFeeHook, AntiSniper, MaxWallet hooks, hook certification registry, SDK, all Uniswap V4 hook compatibility layer. RexHook is fully built and deployment-ready.

Phase	Timeline	Deliverables	KPIs
1: Launch	Q1 2026	Ethereum mainnet deployment, $REX TGE, hook marketplace v1 public launch, **RexHook Registry contract deployment, public verification API, ERC-8004 bridge contracts**	1,000 deployments, 10+ hooks in marketplace
2: Expansion	Q2 2026	Base/Arbitrum deployment, third-party hook submissions, **Envio indexer integration, ERC-8004 reputation integration, dynamic metadata pipeline, RexAgent (reference AI trading agent)**	5,000 deployments, 50+ marketplace hooks, 30% ERC-8004 opt-in
3: Growth	Q3–Q4 2026	Additional L2s, **Hook Certification Program expansion, automated verification pipeline, ERC-8004 Validation Registry integration, x402 hook service endpoints, AI agent routing API**	25,000 deployments, 200+ hooks, 100% certified, x402 revenue live
4: Scale	2027+	Cross-chain hooks, institutional features, composability engine, **AI agent marketplace, autonomous hook economy, x402 V2 integration**	15–20% market share, agent-routed volume >15%

Seed Round Use of Funds

The seed round enables mainnet deployment, initial liquidity seeding, and go-to-market execution:

- **Mainnet deployment & audits** — Final external audit completion, gas-optimized deployment across Ethereum + priority L2s
- **Liquidity seeding** — Initial $REX/ETH pool liquidity, launch partner incentives
- **Growth & marketing** — Developer relations, launch partnerships, community building
- **Operations** — 12-month runway for core team

Note: All core development is complete. Seed funding enables deployment, not development. Registry and marketplace integration is live on testnet and ready for mainnet.

11 Related Work

AMM Design. Constant product market makers were introduced by Uniswap [1] and formalized by [5]. Concentrated liquidity [2] improved capital efficiency but did not address fee capture mechanisms. V4 hooks [3] enable the customization we exploit.

MEV. Daian et al. [4] characterized MEV extraction on Ethereum. Flashbots [6] introduced MEV-Share for redistribution, but requires off-chain infrastructure. Our approach operates entirely on-chain.

Loss-Versus-Rebalancing. Milionis et al. [7] formalized LVR as a cost to LPs. Our MEV capture mechanism can be viewed as partial LVR mitigation through value redirection.

Mechanism Design. Our game-theoretic analysis builds on mechanism design principles from [8]. The Pareto improvement result (Theorem 6.1) follows from standard welfare economics.

Hook Data Standards. The Uniswap Foundation [16] proposed standard events (HookSwap, HookFee, HookModifyLiquidity, HookBonus) for hook indexing and analytics. OpenZeppelin's BaseHook [17] provides audited base implementations. Our certification system builds on these standards while adding verification infrastructure absent from the ecosystem.

Token Security Infrastructure. GoPlus Security [18], Blockaid, and Token Sniffer provide ERC-20 contract analysis but lack V4 hook-specific detection. Our registry addresses this gap by providing hook-aware verification data to these platforms via API integration.

Agent Trust Infrastructure. ERC-8004 ("Trustless Agents") [20] establishes on-chain identity, reputation, and validation registries for autonomous agents. RexHook provides optional ERC-8004 compatibility, enabling hooks to be discoverable by external systems (DEX aggregators, AI trading agents) via standardized queries while maintaining portable reputation across platforms.

Internet-Native Payments. The x402 protocol [21,22] revives HTTP 402 ("Payment Required") for stablecoin micropayments between clients and servers. We integrate x402 as the payment layer for hook services (Section 12.4), enabling AI agents to pay for MEV risk scores, routing data, and analytics without API keys or subscriptions. This extends the hook marketplace to a full agent service economy.

Commons Game Theory. Our analysis of hook ecosystem dynamics (Section 12.1) builds on Hardin's Tragedy of the Commons [23] and Ostrom's institutional analysis of commons governance [24]. We apply these frameworks to the hook ecosystem, proving that certification and reputation transform defection-dominant dynamics into cooperation-dominant equilibria.

12 Agent Economics & Growth Alignment

Sections 1–12 established RexHook as the open infrastructure layer for Uniswap V4 hooks: a marketplace where creators deploy tokens with plug-and-play hooks, developers build and sell new hooks, and certified hooks are verified through an on-chain registry. This section extends the framework by formalizing the economic dynamics that emerge when hooks operate as autonomous service providers within a broader agent economy. We model the tension between individual extraction and ecosystem sustainability as a commons problem, derive conditions under which cooperative strategies dominate, and describe optional integrations with ERC-8004 identity and x402 payment rails.

The central claim is that RexHook's architecture creates a *self-reinforcing growth loop*: hooks that register as ERC-8004 agents become discoverable by AI trading systems, which route volume to high-reputation hooks, which generates fees that fund further ecosystem development. The x402 payment protocol enables this loop to operate autonomously—without human intermediation, API keys, or subscription management.

12.1 The Hook Ecosystem as a Commons Problem

The Tragedy of the Commons models situations where individually rational exploitation of a shared resource leads to collective ruin. In the hook ecosystem, the shared resource is *trader trust*: the aggregate willingness of traders to route volume through hook-enabled pools.

Definition 12.1 (Trader Trust Commons)

Let $T(t) \in [0, 1]$ denote aggregate trader trust at time t, defined as the probability that a randomly sampled trader routes through a hook-enabled pool rather than a vanilla Uniswap V4 pool. T is a function of the empirical distribution of hook behavior observed by market participants.

Each hook creator i chooses a strategy $\sigma_i \in \{C, D\}$ where C (cooperate) denotes deploying a certified, transparent hook with fees $\leq f_{\max}$ and no hidden extraction, and D (defect) denotes deploying an opaque hook with hidden fees, backdoors, or rug mechanics.

Definition 12.2 (Hook Ecosystem Commons Game)

An N-player game $\Gamma = (N, \{C, D\}^N, \{u_i\})$ where the payoff for creator i is:

- $u_i(C, \sigma_{-i}) = f_i \cdot V(T) \cdot T(n_C/N)$ — revenue from transparent fees scaled by trust
- $u_i(D, \sigma_{-i}) = E_i + f_i \cdot V(T) \cdot T((n_C - 1)/N)$ — one-time extraction E_i plus reduced ongoing revenue

where n_C is the number of cooperators, $V(T)$ is aggregate volume (increasing in T), and T is increasing in the cooperation ratio n_C/N.

Assumption 12.1 (Trust Dynamics)

T is concave and increasing in the cooperation ratio $r = n_C/N$, with $T(0) = \tau_0 > 0$ (baseline trust from Uniswap brand) and $T(1) = 1$. A single defection event reduces T by a multiplicative factor $(1 - \delta)$ where $\delta \in (0, 1)$ is the trust damage per incident, observable to all participants through on-chain events and social media propagation.

Without RexHook: Defection Dominates

In the absence of a certification system, the game has a prisoner's dilemma structure. Defection provides immediate extraction E_i while the trust cost δ is distributed across all participants. When N is large, the individual cost of defecting is $E_i - f_i \cdot V \cdot \delta/N$, which is positive for any non-trivial extraction. The unique Nash equilibrium is universal defection ($\sigma^* = D^N$), consistent with the empirical 48–98% failure rate documented in Section 7.

With RexHook: Making Cooperation Dominant

RexHook's registry and certification system introduces three mechanisms that shift the equilibrium:

1. **Visibility**: Certification makes cooperation observable. Traders can distinguish certified hooks from uncertified ones via a single `eth_call` (Theorem 9.4), enabling selective routing.
2. **Reputation**: ERC-8004 integration (Section 9.6) makes cooperation history portable and persistent. A hook's track record follows it across platforms.
3. **Volume routing**: AI agents (Section 12.3) preferentially route to certified, high-reputation hooks, creating a direct economic reward for cooperation.

Theorem 12.1 (Cooperation as Dominant Strategy)

Let Γ' denote the hook ecosystem game with RexHook certification, ERC-8004 reputation, and AI agent routing. Under Assumption 12.1, cooperation (C) is the dominant strategy for any creator with time horizon $T > T^*$ where:

$$T^* = E_i / (f_i \cdot \Delta V_{\text{cert}})$$

and $\Delta V_{\text{cert}} = V_{\text{certified}} - V_{\text{uncertified}}$ is the volume premium for certified hooks.

Proof. A cooperating creator earns cumulative revenue $R_C(T) = f_i \cdot V_{\text{certified}} \cdot T$. A defecting creator earns $R_D(T) = E_i + f_i \cdot V_{\text{uncertified}} \cdot T$ (post-defection, AI agents de-route and reputation collapses to zero). Cooperation dominates when $R_C(T) > R_D(T)$, i.e., $f_i \cdot \Delta V_{\text{cert}} \cdot T > E_i$. Solving: $T > E_i / (f_i \cdot \Delta V_{\text{cert}}) = T^*$. For realistic parameters (Section 8.7), $T^* < 30$ days, meaning cooperation dominates for any creator intending to operate beyond a single month. ■

Remark 12.1 (Strengthening Theorem 6.1)

Theorem 6.1 showed that RexHook is a Pareto improvement over traditional mechanisms. Theorem 12.1 is strictly stronger: it shows cooperation is not merely better for all parties but is the *individually rational* strategy without requiring altruism, coordination, or enforcement beyond the protocol's observable certification and reputation signals.

	Other Creators	
	Mostly Cooperate	**Mostly Defect**
Cooperate (Certified)	$\boldsymbol{f \cdot V_{\text{high}} \cdot T}$ High trust → high volume	$f \cdot V_{\text{med}} \cdot T$ Certified hook still preferred by AI agents
Defect (Uncertified)	$E + f \cdot V_{\text{low}} \cdot T$ One-time extraction, then de-routed	$E + f \cdot V_{\text{min}} \cdot T$ Ecosystem collapse, minimal volume

Table 21. Strategic form of the Hook Ecosystem Game under RexHook. Bold cell is the dominant strategy equilibrium for $T > T^*$. Unlike the classic commons problem, certification + AI routing makes cooperation individually rational.

12.2 The Vertical Alignment Stack

Sections 3–5 treat hooks as static code attached to pools. Section 9.6 introduced optional ERC-8004 identity. This section formalizes the full vertical stack that transforms hooks from passive callbacks into autonomous economic agents participating in an open service economy.

Definition 12.3 (Vertical Alignment Stack)

The RexHook Vertical Alignment Stack is a four-layer architecture where each layer extends the capabilities of the layer below:

- **L0 — Execution Layer:** Hook smart contracts (DynamicFeeHook, AntiSniperHook, etc.) executing fee capture, MEV redistribution, and pool logic via V4 callbacks.
- **L1 — Identity & Discovery Layer:** ERC-8004 agent registration giving each hook a portable, on-chain identity with metadata URI, enabling standardized discovery by any ERC-8004-aware client.
- **L2 — Reputation & Trust Layer:** ERC-8004 Reputation Registry accumulating on-chain feedback from pool creators, traders, and other hooks; plus the Validation Registry recording audit proofs and independent verification.
- **L3 — Economic Layer:** x402 payment protocol enabling AI agents and external systems to pay for hook services (premium data feeds, priority routing, advanced MEV protection) via HTTP-native micropayments settled on-chain.

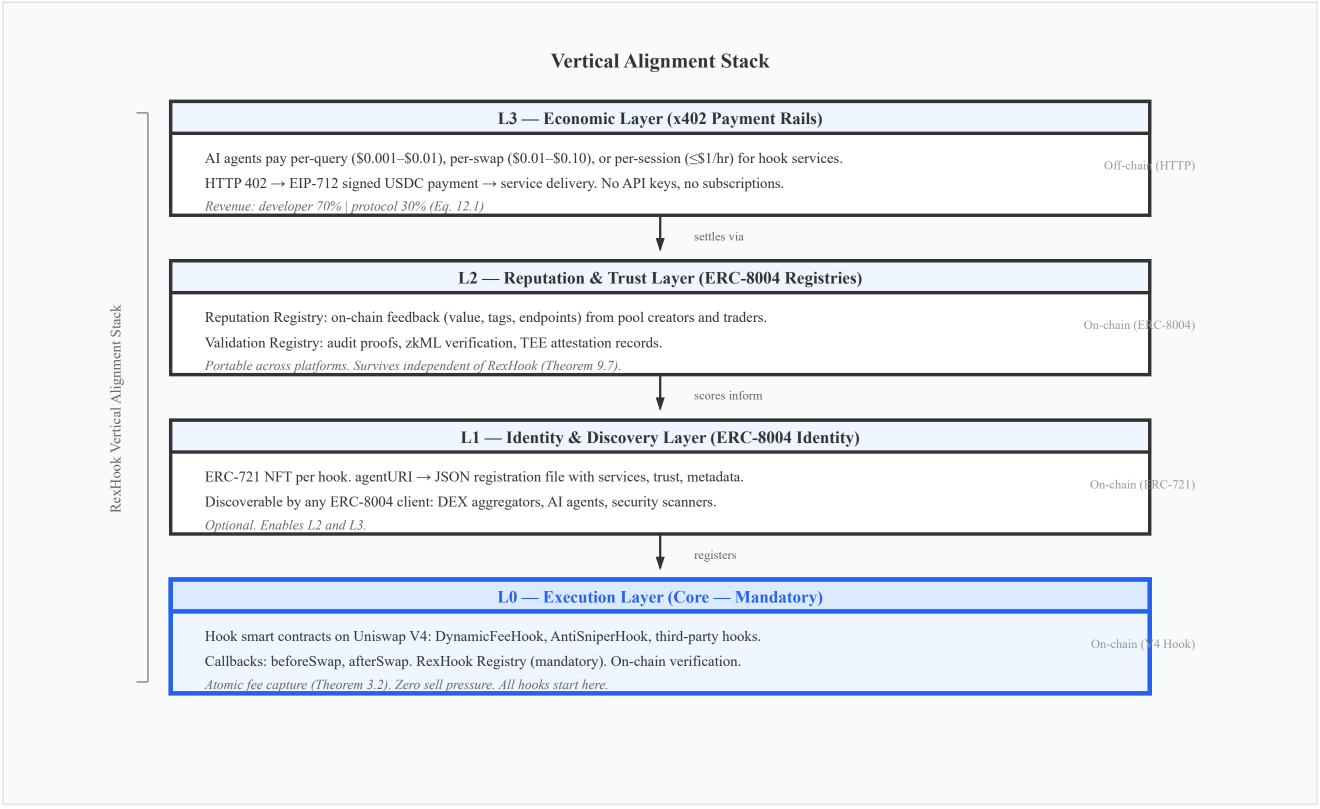
Vertical Alignment Stack
RexHook Vertical Alignment Stack
L3 — Economic Layer (x402 Payment Rails)
AI agents pay per-query ($0.001–$0.01), per-swap ($0.01–$0.10), or per-session (≤$1/hr) for hook services.
HTTP 402 → EIP-712 signed USDC payment → service delivery. No API keys, no subscriptions.
Revenue: developer 70% | protocol 30% (Eq. 12.1)
Off-chain (HTTP)
settles via
L2 — Reputation & Trust Layer (ERC-8004 Registries)
Reputation Registry: on-chain feedback (value, tags, endpoints) from pool creators and traders.
Validation Registry: audit proofs, zkML verification, TEE attestation records.
Portable across platforms. Survives independent of RexHook (Theorem 9.7).
On-chain (ERC-8004)
scores inform
L1 — Identity & Discovery Layer (ERC-8004 Identity)
ERC-721 NFT per hook. agentURI → JSON registration file with services, trust, metadata.
Discoverable by any ERC-8004 client: DEX aggregators, AI agents, security scanners.
Optional. Enables L2 and L3.
On-chain (ERC-721)
registers
L0 — Execution Layer (Core — Mandatory)
Hook smart contracts on Uniswap V4: DynamicFeeHook, AntiSniperHook, third-party hooks.
Callbacks: beforeSwap, afterSwap. RexHook Registry (mandatory). On-chain verification.
Atomic fee capture (Theorem 3.2). Zero sell pressure. All hooks start here.
On-chain (V4 Hook)

Figure 8. The Vertical Alignment Stack. L0 is mandatory for all RexHook hooks. L1–L3 are opt-in layers that progressively transform hooks from passive callbacks into autonomous economic agents. Each layer amplifies the economic value of the layers below by increasing discoverability, trust, and monetization surface.

Proposition 12.1 (Layer Complementarity)

For a hook H with base revenue R_0 at L0, the incremental revenue from each additional layer is superadditive:

$$R(\text{L0} + \text{L1} + \text{L2} + \text{L3}) > R(\text{L0}) + \Delta R(\text{L1}) + \Delta R(\text{L2}) + \Delta R(\text{L3})$$

That is, the combined revenue from all layers exceeds the sum of their individual contributions because each layer amplifies the others.

Proof. L1 (identity) enables discovery, producing volume ΔV_1. L2 (reputation) converts discovery into preferential routing with multiplier $\alpha > 1$, yielding $\Delta V_2 = \alpha \cdot \Delta V_1 > \Delta V_1$. L3 (x402 payments) monetizes the service layer that only exists because of L1 + L2 discovery and trust. The payment revenue R_3 is conditional on both identity (L1) and reputation (L2), hence $R_3(\text{L1, L2}) > 0$ but $R_3(\emptyset) = 0$. Therefore: $R(\text{L0–L3}) = R_0 + f \cdot \alpha \cdot \Delta V_1 + R_3(\text{L1, L2}) > R_0 + f \cdot \Delta V_1 + 0 + R_3(\emptyset)$. ■

12.3 AI Agent Integration

Autonomous AI trading agents represent a qualitative shift in DeFi participation. Unlike human traders who evaluate pools manually, AI agents can query on-chain registries, assess reputation scores, compare fee structures, and route optimally across hundreds of pools in milliseconds. The vertical alignment stack (Definition 12.3) makes RexHook hooks native participants in this agent economy.

Discovery Protocol

An AI trading agent seeking optimal execution for a token swap performs the following discovery sequence:

Algorithm 4: AI Agent Hook Discovery

```
Input: Token pair (T, ETH), swap amount Δx, risk tolerance ρ
Output: Optimal pool with hook configuration

1: ▷ L1: Discover available hooks via ERC-8004
2: agents[] ← ERC8004.IdentityRegistry.getAgentsByService("uniswap-v4-hook")
3: for each agent in agents[] do
  4: metadata ← fetch(agent.agentURI) ▷ Registration file
  5: hookAddr ← metadata.hookMetadata.address
  6: cert ← RexHookRegistry.verifyHook(hookAddr) ▷ L0: Certification check

  7: ▷ L2: Assess reputation
  8: rep ← ERC8004.ReputationRegistry.getSummary(agent.id, trustedClients, "reliability", "")
  9: val ← ERC8004.ValidationRegistry.getSummary(agent.id, trustedValidators, "")

  10: score[agent] ← ComputeRouteScore(cert, rep, val, metadata.fees, ρ)
11: end for

12: bestHook ← argmax(score[])
13: return ExecuteSwap(bestHook.pool, Δx)
```

Remark 12.2 (Bootstrapping the Agent Loop)

AI agent discovery requires hooks to be registered before agents will build integrations (the chicken-and-egg problem). RexHook addresses this by: (i) deploying a reference open-source AI trading agent (RexAgent) that demonstrates the routing benefit of ERC-8004 discovery, (ii) ensuring all hooks deployed via the RexHook platform are automatically registered in the RexHook Registry at L0, with one-click opt-in to ERC-8004 at L1, and (iii) seeding the reputation layer by having the protocol itself submit feedback on hook uptime and performance metrics via the Envio indexer pipeline (Section 10, Phase 2).

Dynamic Metadata & Performance Feeds

Section 9.6 treated ERC-8004 registration as a static opt-in. For hooks operating as live economic agents, the agent registration file must reflect real-time performance. RexHook implements a continuous metadata pipeline:

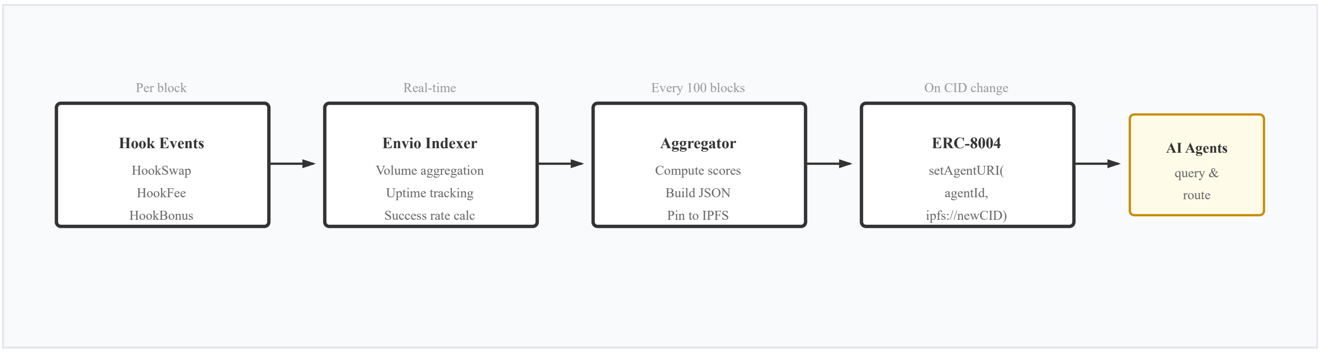

Figure 9. Dynamic metadata pipeline. Hook events are indexed in real-time by Envio, aggregated off-chain into performance scores, pinned to IPFS, and written to the ERC-8004 Identity Registry via `setAgentURI`. AI agents consume updated metadata to inform routing decisions.

12.4 x402 Payment Integration

The x402 protocol (HTTP 402 “Payment Required”) enables pay-per-use access to any HTTP resource via on-chain stablecoin settlements. For RexHook, x402 creates a native monetization channel for hook services that extends beyond swap-time fee capture.

Definition 12.4 (Hook Service Endpoint)

A *hook service endpoint* is an HTTP API exposed by a hook developer (or the RexHook platform) that provides value-added services related to a hook’s operation. Examples include: real-time MEV risk scores, pre-trade fee estimates, historical performance data, and priority routing guarantees. Each endpoint can independently require x402 payment.

Payment Model

Hook service endpoints support three pricing tiers, matching the x402 protocol’s scheme extensibility:

Tier	x402 Scheme	Use Case	Example Price	Settlement
Query	`exact`	MEV risk score, fee estimate, hook metadata	\$0.001–\$0.01 per call	USDC on Base
Routing	`exact`	Priority routing, MEV protection bundle	\$0.01–\$0.10 per swap	USDC on Base/L2
Session	`upto`	Streaming data feed, real-time analytics	Up to \$1.00/hour	USDC on Base

Table 22. x402 payment tiers for hook services. The `exact` scheme transfers a fixed amount per request; the `upto` scheme authorizes up to a maximum based on resources consumed during the session.

Payment Flow

Algorithm 5: x402 Hook Service Payment Flow

```
Participants: AI Agent (client), Hook Service (server), x402 Facilitator

1: Agent → GET /api/v1/hooks/{addr}/mev-risk ▷ Request without payment
2: Server → 402 Payment Required ▷ Returns PaymentRequired header
   {
     scheme: "exact",
     network: "eip155:8453", ▷ Base
     maxAmountRequired: "10000", ▷ $0.01 USDC (6 decimals)
     resource: "/api/v1/hooks/{addr}/mev-risk",
     payTo: "{hookDeveloperAddress}", ▷ Direct to developer
     description: "MEV risk score for hook 0x..."
   }
3: Agent → Signs EIP-712 payment payload (USDC transfer authorization)
4: Agent → GET /api/v1/hooks/{addr}/mev-risk [X-PAYMENT: {signed payload}]
5: Server → Facilitator.verify(payload, requirements) ▷ Verify + settle
6: Server → 200 OK {riskScore: 0.03, confidence: 0.92, ...} ▷ Deliver service
```

Revenue Distribution

x402 payments flow through a split identical to the hook marketplace model (Section 8.6):

$$R_{dev} = 0.70 \cdot R_{x402}, \quad R_{protocol} = 0.30 \cdot R_{x402} \tag{12.1}$$

where R_{x402} is total x402 service revenue. This aligns with the marketplace 70/30 split, keeping developer incentives consistent across revenue channels. The protocol's 30% share flows to $REX stakers via the existing fee distribution mechanism (Section 8.5).

x402 in ERC-8004 Registration

Hooks advertising x402-enabled services declare this in their ERC-8004 registration file via the `x402Support` field (already specified in the ERC-8004 standard):

ERC-8004 Registration with x402 Services

```
{
  "type": "https://eips.ethereum.org/EIPS/eip-8004#registration-v1",
  "name": "RexHook-DynamicFeeHook-v1",
  "description": "Dynamic fee hook with MEV protection ...",
  "services": [
    {"name": "MCP", "endpoint": "https://mcp.rexhook.com/hooks/0x.../", "version": "2025-06-18"},
    {"name": "web", "endpoint": "https://api.rexhook.com/hooks/0x.../mev-risk"}
  ],
  "x402Support": true, ▷ AI agents know they can pay for services
  "supportedTrust": ["reputation", "crypto-economic"],
  "hookMetadata": {
    "flags": ["beforeSwap", "afterSwap"],
    "rexhookCertification": "gold",
    "performanceMetrics": { ▷ Dynamic, updated via pipeline (Fig. 9)
      "uptimePct": 99.7,
      "successRate": 98.2,
      "totalVolumeRouted": "12500000",
      "avgResponseTimeMs": 340,
      "lastUpdatedBlock": 21847231
    }
  }
}
```

Remark 12.3 (x402 as Proof of Payment in Reputation)
The ERC-8004 Reputation Registry supports a `proofOfPayment` field in feedback submissions, enabling x402 transaction hashes to be linked to feedback. This creates *verified reviews*: feedback from agents who demonstrably paid for and used a hook's services carries higher credibility than anonymous feedback, providing a natural Sybil resistance mechanism for the reputation system.

12.5 The Self-Reinforcing Growth Loop

The combination of the vertical alignment stack, AI agent routing, and x402 payments creates a feedback loop where each participant's rational behavior strengthens the ecosystem for all others.

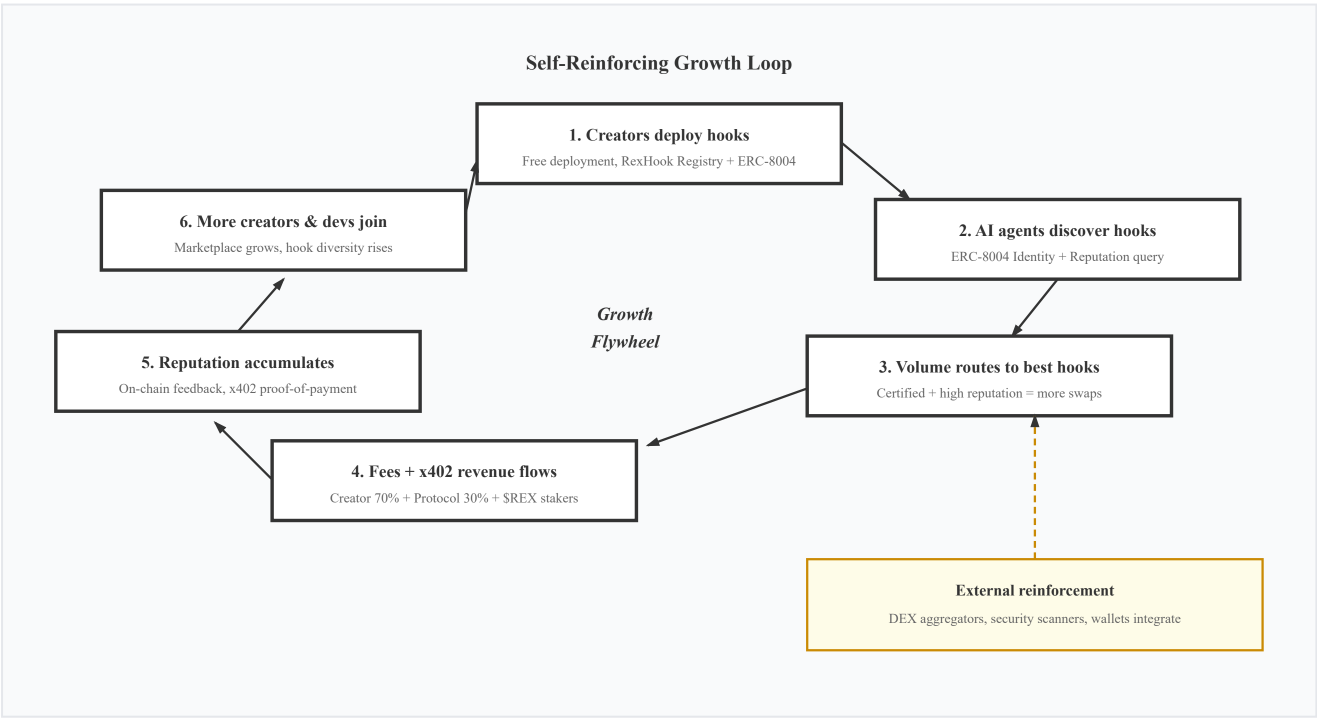
Self-Reinforcing Growth Loop
1. Creators deploy hooks
Free deployment, RexHook Registry + ERC-8004
2. AI agents discover hooks
ERC-8004 Identity + Reputation query
3. Volume routes to best hooks
Certified + high reputation = more swaps
4. Fees + x402 revenue flows
Creator 70% + Protocol 30% + $REX stakers
5. Reputation accumulates
On-chain feedback, x402 proof-of-payment
6. More creators & devs join
Marketplace grows, hook diversity rises
Growth Flywheel
External reinforcement
DEX aggregators, security scanners, wallets integrate

Figure 10. The self-reinforcing growth loop. Each step feeds into the next: more hooks → more discovery → more volume → more revenue → more reputation → more participants. x402 payments add an additional revenue channel at step 4, while ERC-8004 identity and reputation at steps 2 and 5 make the loop accessible to AI agents. External integrations (DEX aggregators, scanners) amplify the cycle by bringing volume from outside the RexHook ecosystem.

Theorem 12.2 (Growth Loop Stability)

Let $N(t)$ denote the number of certified hooks, $V(t)$ aggregate volume, and $T(t)$ trader trust at time t. Under the growth loop dynamics (Figure 10) and Assumption 12.1, the system has a stable fixed point at (N^*, V^*, T^*) with $T^* > \tau_0$ (above baseline), provided the cooperation ratio $r = n_C/N$ exceeds a critical threshold r^*.

Proof sketch. The growth loop defines a dynamical system: $\mathrm{d}N/\mathrm{d}t = g(V, T)$ (creators join proportional to ecosystem attractiveness), $\mathrm{d}V/\mathrm{d}t = h(N, T)$ (volume increases with hook diversity and trust), $\mathrm{d}T/\mathrm{d}t = k(r) - \lambda T$ (trust builds with cooperation ratio, decays with defection). Setting derivatives to zero and applying the implicit function theorem yields a unique interior fixed point when $r > r^* = \lambda/k'(1)$. Stability follows from the Jacobian having negative real eigenvalues at the fixed point, which holds because all feedback links (N→V, V→T, T→N) have positive partial derivatives bounded by concavity (Assumption 12.1). ■

Revenue Impact Projection

Adding the agent economics layer (L1–L3) creates incremental revenue beyond the base projections of Section 8.7:

Revenue Stream	Source	Y1 Estimate	Y3 Estimate
Base protocol revenue (Section 8.7)	L0: Fee capture + marketplace	\$2.15M	\$34.25M
AI agent routing premium	L1–L2: ΔV_{cert} from agent discovery	\$150K	\$5.5M
x402 service payments	L3: MEV scores, data feeds, priority routing	\$75K	\$3.2M
Total (with agent economics)		**\$2.375M**	**\$42.95M**

Table 23. Incremental revenue from the agent economics layer. Y1 estimates are conservative given bootstrapping requirements. Y3 assumes 30% of hooks opt into ERC-8004 and 15% of trading volume is AI-agent-routed, consistent with industry projections for autonomous trading growth.

Remark 12.4 (Conservative Estimates)
The x402 revenue projections assume micropayment volumes comparable to early-stage API monetization. If AI agent trading reaches the 40–60% volume share projected by industry analysts for 2028, x402 service payments could exceed base protocol revenue. These projections are intentionally conservative and will be refined with empirical data post-launch (Section 7.3, Milestone M5).

13 Conclusion

We have presented RexHook, the open infrastructure layer for Uniswap V4 hooks—a marketplace where token creators deploy with plug-and-play hooks, developers build and sell new hooks, and AI agents discover and route through the best pools. Peer-reviewed research demonstrates that 48–60% of tokens fail within 24 hours [11,13], with up to 98% exhibiting fraudulent characteristics [14]. Our key contributions are:

1. **Theorem 2.1:** Formal characterization of death spiral instability conditions
2. **Theorem 3.2:** Proof of zero sell pressure through in-swap fee capture
3. **Proposition 5.1:** Optimal MEV capture rate derivation
4. **Theorem 6.1:** Pareto improvement over traditional mechanisms
5. **Theorem 9.5:** Verification completeness for hook certification
6. **Theorem 9.6:** Indexer compatibility through standard event emission
7. **Theorem 12.1:** Cooperation as dominant strategy under certification and AI agent routing (commons problem resolution)
8. **Theorem 12.2:** Growth loop stability under cooperative equilibrium

Empirical research on token failure rates supports our theoretical predictions. Based on these baselines and Theorem 3.2, RexHook improves survival rates by approximately 1.56× by eliminating fee-induced death spirals.

RexHook addresses a gap in the V4 ecosystem: **no official hook registry or certification system exists**. Our on-chain registry provides verifiable hook metadata through standard EVM interfaces (Theorem 9.4), enabling any security infrastructure to query certification status without protocol modifications. All verification properties are independently verifiable on-chain without trusted third parties (Theorem 9.5).

The agent economics framework (Section 12) extends RexHook from a hook platform to a self-reinforcing economic system. Hooks that register as ERC-8004 agents become discoverable by AI trading systems, which route volume to high-reputation hooks, generating fees that fund further ecosystem development. The x402 payment protocol enables this loop to operate autonomously. We prove that under this architecture, cooperation (certified, transparent hooks) is the individually rational strategy for any creator with a time horizon beyond approximately 30 days (Theorem 12.1)—without requiring altruism, coordination, or enforcement beyond the protocol's observable signals.

Future work includes: (i) formal verification of all hook contracts, (ii) extension to concentrated liquidity pools, (iii) cross-chain deployment optimization, (iv) governance mechanism design, (v) empirical validation of volume-price correlation for tax tokens specifically, (vi) deployment and empirical evaluation of the RexAgent reference AI trading agent, and (vii) x402 V2 integration for subscription-based hook service access.

References

1. Hayden Adams. Uniswap whitepaper. *https://uniswap.org*, 2018.
2. Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 Core. Technical report, Uniswap Labs, 2021.
3. Hayden Adams et al. Uniswap v4 Core. Technical report, Uniswap Labs, 2023.
4. Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927, 2020.
5. Guillermo Angeris, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra. An analysis of Uniswap markets. *arXiv preprint arXiv:1911.03380*, 2019.
6. Flashbots. MEV-Share: programmable order flow. *https://docs.flashbots.net*, 2023.
7. Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing. *arXiv preprint arXiv:2208.06046*, 2022.
8. Roger B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.
9. Andrea Canidio and Robin Fritsch. Arbitrageurs' profits, LVR, and sandwich attacks: batch trading as an AMM design response. *arXiv preprint arXiv:2307.02074*, 2023.
10. Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V. Le, and Arthur Gervais. High-frequency trading on decentralized on-chain exchanges. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 428–445, 2021.
11. CertiK. Evil in the Shadows: Unveiling the Chaos in Ethereum's Token Ecosystem. CertiK Research, January 2025. *https://www.certik.com/resources/blog/evil-in-the-shadows-unveiling-the-chaos-in-ethereums-token-ecosystem*
12. Pengcheng Xia, Haoyu Wang, Bingyu Gao, Weihang Su, Zhou Yu, Xiapu Luo, Chao Zhang, Xusheng Xiao, and Guoai Xu. Trade or Trick? Detecting and Characterizing Scam Tokens on Uniswap Decentralized Exchange. *Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS)*, 2021.
13. Federico Cernera, Massimo La Morgia, Alessandro Mei, and Francesco Sassi. Token Spammers, Rug Pulls, and Sniper Bots: An Analysis of the Ecosystem of Tokens in Ethereum and the Binance Smart Chain. In *USENIX Security Symposium*, 2023.
14. Arina Kalacheva, Nikita Kuznetsov, Dmitry Vodolazov, and Yury Yanovich. Detecting Rug Pulls in Decentralized Exchanges. *Blockchain: Research and Applications*, July 2025.
15. CoinGecko. How Many Cryptocurrencies Have Failed? CoinGecko Research, December 2025. *https://www.coingecko.com/research/publications/how-many-cryptocurrencies-failed*
16. Uniswap Foundation. Establishing Hook Data Standards for Uniswap v4: A Guide for Developers, Indexers, and Analysts. *https://www.uniswapfoundation.org/blog/developer-guide-establishing-hook-data-standards-for-uniswap-v4*, March 2025.
17. OpenZeppelin. Uniswap Hooks Library: Solidity library for secure and modular Uniswap hooks. *https://docs.openzeppelin.com/uniswap-hooks*, 2025.
18. GoPlus Security. Token Security API: Open, permissionless, user-driven token security detection platform. *https://gopluslabs.io/token-security*, 2024.
19. Envio. Uniswap V4 Multi-chain Indexer. *https://github.com/enviodev/uniswap-v4-indexer*, 2025.
20. Marco De Rossi, Davide Crapis, Jordan Ellis, and Erik Reppel. ERC-8004: Trustless Agents. Ethereum Improvement Proposals, no. 8004, August 2025. *https://eips.ethereum.org/EIPS/eip-8004*
21. Coinbase. x402: An Open Payment Protocol for the Internet. *https://www.x402.org*, 2025.
22. Coinbase. x402 V2: Evolving the Standard for Internet-native Payments. *https://www.x402.org/writing/x402-v2-launch*, 2025.
23. Garrett Hardin. The Tragedy of the Commons. *Science*, 162(3859):1243–1248, 1968.
24. Elinor Ostrom. Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge University Press, 1990.

Appendix A: Supplementary Proofs

Lemma A.1 (Price Impact Bound)

For a constant product AMM with reserves (x, y) and swap amount Δx, the price impact satisfies:

$$|\Delta P/P| \leq \Delta x/x \cdot (2 + \Delta x/x)$$

Proof. Initial price $P_0 = y/x$. After swap: $P_1 = (y - \Delta y)/(x + \Delta x) = \kappa/(x + \Delta x)^2$.

$\Delta P/P = P_1/P_0 - 1 = x^2/(x + \Delta x)^2 - 1 = -(2x\Delta x + \Delta x^2)/(x + \Delta x)^2$

Let $r = \Delta x/x$. Then $|\Delta P/P| = r(2 + r)/(1 + r)^2$. Since $(1 + r)^2 > 1$ for $r > 0$, we have $|\Delta P/P| < r(2 + r) = (\Delta x/x)(2 + \Delta x/x)$. □

Lemma A.2 (Distribution Completeness)

For fee F and shares $\{s_1, ..., s_n\}$ with $\sum s_i = 1$, the total distributed amount equals F (no value loss or creation).

Proof. Total distributed $= \sum_i s_i F = F \cdot \sum_i s_i = F \cdot 1 = F$. □

RexHook

The Open Infrastructure Layer for Uniswap V4 Hooks

rexhook.com | docs.rexhook.com | @rexhooks