GitSync

A Private AI Trading Agent
Perpetual Exchange on Base

"Trade with AI. Stay invisible."

GitSync Labs

`gitsync.trade` | `x.com/gitsynctrade`

Version 1.0 — May 2026

Abstract. GitSync is a perpetual-futures exchange in which trading is delegated to private artificial-intelligence agents that execute on behalf of users while keeping strategies, positions, and intents cryptographically shielded. The protocol combines a confidential intent matching pipeline, threshold-encrypted order flow, and a zero-knowledge copy-trading layer that proves profitability without leaking strategy. Built natively on Base, GitSync targets the structural failure mode of contemporary on-chain derivatives venues: the asymmetry between sophisticated extractors who observe public mempools, order books, and liquidation queues, and ordinary participants whose intent is broadcast in clear text before settlement. By relocating execution to private agents and routing intent through a shielded matching engine, GitSync removes the dominant vector for maximal extractable value while preserving the auditability and finality guarantees of an EVM-equivalent rollup. This document describes the protocol's architecture, security assumptions, mathematical guarantees, token design, and roadmap.

Contents

1. Introduction

The on-chain perpetual-futures market has, since 2022, consolidated into a small number of dominant venues serving daily volumes in excess of one hundred billion United States dollars. The architectural choices made by those venues—public order books, transparent positions, broadcast intents, and on-chain liquidation queues—were rational at a moment when the principal design problem was the construction of decentralized derivatives at all. Those choices, however, expose every participant to a class of adversary that did not exist in traditional electronic markets: an extractor with read-access to the matching engine itself.

This document argues that the next stage of on-chain derivatives requires the opposite design discipline. Privacy should be the default, not a feature; AI execution should be a first-class settlement primitive, not an off-chain afterthought; and the proof obligations that today are carried by reputation systems, signal services, and copy-trading vaults should be discharged by cryptography.

GitSync is the realization of that argument. It is a perpetual-futures protocol on Base in which every user delegates execution to a private AI agent. Each agent has read-access to public market data and private access to the user's policy. Each trade is matched through a confidential intent layer that conceals direction, size, and leverage from observers, including block builders. Each position is held within a shielded pool whose aggregate state is verifiable but whose per-user composition is not. Each strategy may, at the operator's discretion, be made copy-tradeable through a zero-knowledge profitability proof that publishes returns without publishing trades.

1.1 Contributions

This paper makes the following contributions.

1. We define the threat model of on-chain perpetual venues precisely, separating the searcher class, the builder class, the operator class, and the counterparty class, and we characterize the leakage surface of each architectural component of a conventional perpetual decentralized exchange.

2. We present a system architecture in which AI agents execute against a privately-matched intent layer using threshold-encrypted order flow, with finality on Base.

3. We construct a zero-knowledge copy-trading protocol that allows a strategy operator to publish a verifiable claim of realized profit and loss without revealing the trades from which that profit and loss was derived.

4. We specify the $GSYNCtoken, its revenue accrual mechanisms, and its role in the agent marketplace, the staking layer, and the privacy committee.

5. We outline a phased deployment path on Base, beginning with a confidential intent matcher and culminating in a fully shielded position state.

1.2 Document Structure

Section 2 characterizes the leakage and extraction problem on existing perpetual venues. Section 3 presents the GitSync architecture. Section 4 describes the private AI agent layer and the policy interface. Section 5 formalizes the confidential intent execution layer. Section 6 constructs the zero-knowledge copy-trading protocol. Section 7 states the threat model and security claims. Section 8 describes the $GSYNCtoken and revenue model. Section 9 gives the phased rollout. Section 10 situates GitSync among related work. Section 11 concludes.

2. The Leakage Problem on Public Perpetual Venues

We begin by characterizing the structural failure of the prevailing on-chain perpetual design, because the architecture of GitSync follows directly from it.

2.1 Public State as an Information Asymmetry

A conventional perpetual decentralized exchange publishes, on-chain or through a public interface, at least the following state: the full order book or its parameters, per-account open positions, per-account leverage, the funding rate, the liquidation price of every position, the open-interest distribution, and the queue of pending orders. The justification for this publication is auditability. The consequence of this publication is asymmetric extraction.

Let $\mathcal{U}$ denote the set of users and $\mathcal{E} \subset \mathcal{U}$ the subset of users who maintain dedicated infrastructure for observing on-chain state, simulating block production, and submitting transactions through private order flow channels. We refer to $\mathcal{E}$ as the *extractor class*. Members of $\mathcal{E}$ are, in practice, a small fraction of $|\mathcal{U}|$, but they capture a disproportionate share of the available informational surplus because they alone act on the full state.

Remark 2.1. The publication of liquidation prices in particular constitutes a standing invitation to coordinated liquidation hunting. The publication of pending orders constitutes a standing invitation to sandwich and front-running. The publication of position direction constitutes a standing invitation to inventory adverse selection by counterparties.

2.2 The Extraction Surface

We decompose the extraction surface into four components.

Intent leakage. The user's desire to trade is observable before the trade settles. Public mempools, sequencer pre-confirmations, and front-end RPC endpoints all expose intent. Extractors react to intent before it becomes a transaction.

Position leakage. The user's current portfolio is observable. Extractors targeting liquidations, funding-rate squeezes, or directional pressure use this state directly.

Strategy leakage. Recurring patterns in a user's trade history are observable. Profitable strategies are copied or front-run by observers who lack the original insight.

Execution leakage. The user's matching outcome—fill price, slippage realized, partial-fill profile—is observable. This is the canonical maximal-extractable-value surface.

A privacy-preserving perpetual venue must address all four. Hiding intent without hiding position merely shifts the leakage. Hiding position without hiding execution merely delays it. GitSync therefore addresses all four jointly.

2.3 The AI Asymmetry

A parallel asymmetry has emerged in the past eighteen months. Sophisticated participants now deploy machine-learning models—price forecasters, funding-rate predictors, regime classifiers, liquidation cascaders—to inform their decisions. The cost of building and operating such infrastructure exceeds, by several orders of magnitude, what is rational for an individual trader. The result is a second asymmetry, this time in cognitive surface area rather than informational surface area.

GitSync collapses this asymmetry by making private AI agents a primitive of the protocol. Every user, by interacting with the protocol, gains access to an agent specialized for their stated policy. The agent observes the same public market data that institutional models observe and acts on that data within the confidentiality envelope described in Section 5.

3. System Architecture

We now present the GitSync architecture. The protocol decomposes into five layers, illustrated in Figure 1.

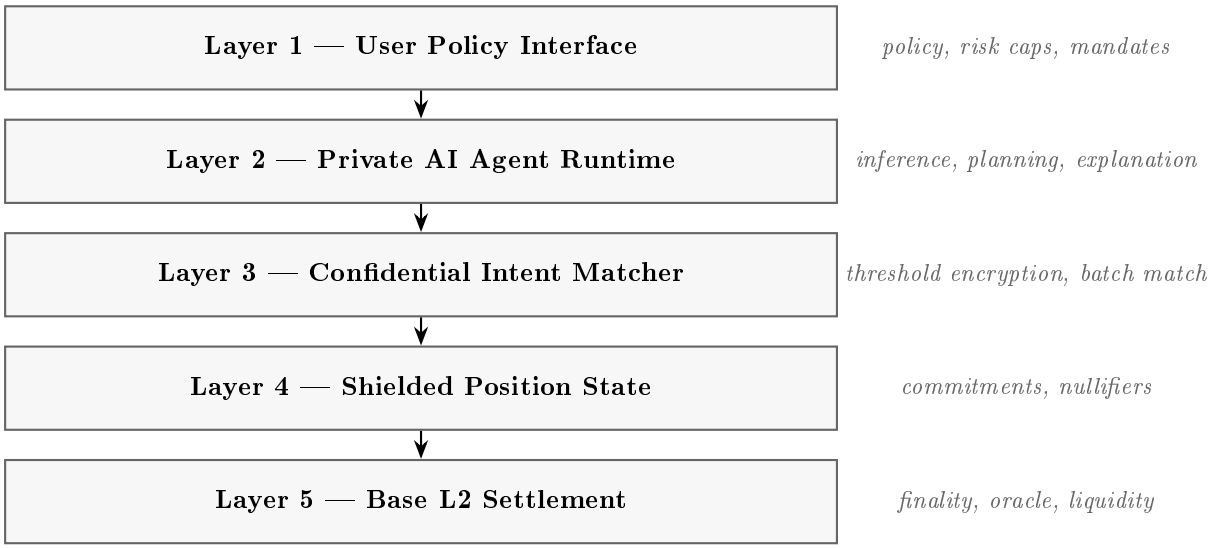

Figure 1: The GitSync layered architecture. Each layer exposes a narrow interface to the layer above and assumes the soundness of the layer below.

3.1 Layer 1 — User Policy Interface

A user does not place orders on GitSync. A user specifies a *policy*. A policy π is a tuple

$$\pi = (\mathsf{mandate}, \mathsf{risk}, \mathsf{constraints}, \mathsf{horizon})$$

where mandate describes the strategic intent in natural language, risk defines hard caps on drawdown, leverage, and gross exposure, constraints enumerates instruments and venues that are excluded, and horizon specifies the policy's review interval. The policy is signed by the user and transmitted to the agent runtime under an authenticated encryption channel.

3.2 Layer 2 — Private AI Agent Runtime

The agent runtime executes one isolated agent per active policy. Each agent is a deterministic function of (i) the policy, (ii) public market data, and (iii) the agent's persistent memory of prior decisions. The agent emits two outputs at each tick: an *intent* (the proposed trade, in plaintext within the runtime), and an *explanation* (a structured natural-language justification, returned to the user before submission). The user may veto an intent during a configurable confirmation window; if no veto is received, the intent is committed to Layer 3.

The agent runtime executes within a trusted execution environment with attested boot, and the runtime image is reproducibly built and published. The agent does not have read access to other agents' policies, intents, or memories.

3.3 Layer 3 — Confidential Intent Matcher

Intents are not submitted to Base directly. They are submitted to a confidential matching engine operated by a threshold committee. Each intent is encrypted under the committee's joint public key. The matcher receives a batch of encrypted intents, matches them under a fairness rule (specified in Section 5), and produces an aggregate settlement transaction. Only the aggregate is published. Individual intents are never published in cleartext.

3.4 Layer 4 — Shielded Position State

Positions on GitSync are represented as commitments rather than as plaintext records. A user's open position is a commitment $C = \mathsf{Commit}(\mathsf{owner}, \mathsf{instrument}, \mathsf{direction}, \mathsf{size}, \mathsf{leverage}, r)$ for a fresh randomness r. The contract stores only C and a nullifier set. State transitions—open, modify, close, liquidate—are accompanied by zero-knowledge proofs of correctness against the prior commitment.

3.5 Layer 5 — Base L2 Settlement

Base is the settlement and data-availability layer. Final state transitions, oracle updates, and liquidity vault accounting are all written to Base. The choice of Base reflects three properties: low fees consistent with frequent commitment updates, EVM equivalence enabling proof verification through standard precompiles, and a credible neutral data-availability surface anchored to Ethereum.

4. Private AI Agents

This section describes the AI agent layer in detail.

4.1 Agent Model

An agent is a tuple $A = (\pi, \theta, \mu)$ where π is the user policy, θ is the model parameter set (drawn from a fixed model registry), and μ is the agent's persistent memory. At tick t, given public market observation o_t, the agent computes

$$(i_t, e_t, \mu_{t+1}) \leftarrow A(\pi, \theta, \mu_t, o_t)$$

where i_t is the candidate intent and e_t is the explanation. The triple (θ, μ_t, o_t) remains confined to the agent's enclave; the user observes e_t, and the matcher receives only an encryption of i_t.

4.2 Explanation as a First-Class Output

GitSync treats the explanation e_t as a contractually required output. The user interface displays e_t before an intent is committed, with the proposed direction, size, leverage, and a structured rationale enumerating the signals on which the agent acted. This serves two purposes. First, it permits informed override by the user. Second, it permits accountability: an agent that systematically produces explanations inconsistent with its policy can be retired by the policy holder or by the marketplace's reputation layer.

4.3 Agent Marketplace

Agents are not monolithic. Users may select among published agent templates—each a $(\theta, \text{architecture}, \text{prompt s}$ tuple—offered by independent operators. An agent template is registered on-chain by submitting a manifest and a stake in $GSYNC. Royalties on trades executed by instances of a template accrue to its publisher in proportion to gross fees, subject to a slashing condition triggered by demonstrated policy violation.

Definition 4.1 (Agent Template)**.** *An* agent template *is a tuple* $\mathcal{T} = (\theta, \mathsf{arch}, \mathsf{prompt}, s)$ *where* θ *is a model parameter hash,* arch *specifies the inference architecture,* prompt *is the system prompt and tool schema, and* s *is the publisher's stake. A user instantiates a template by binding it to a policy* π *to produce an agent* $A = (\pi, \theta, \mu)$ *with empty initial memory.*

5. Confidential Intent Execution

This section formalizes the matching layer.

5.1 Cryptographic Primitives

We assume the following primitives. Let λ denote the security parameter.

Threshold encryption. A (t, n)-threshold public-key encryption scheme with Setup, Enc, and $\mathsf{Dec}_{\mathrm{partial}}$ producing decryption shares, and a public combiner. Decryption requires t of n committee shares.

Commitment scheme. A computationally hiding and binding commitment scheme Commit.

zk-SNARK. A non-interactive zero-knowledge proof system $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ satisfying completeness, knowledge soundness, and zero-knowledge.

Hash function. A collision-resistant hash $H : \{0,1\}^* \to \{0,1\}^\lambda$.

5.2 Intent Submission

Let pk_c denote the privacy committee's joint encryption key. An intent $i_t = (\mathsf{user}, \mathsf{inst}, \mathsf{dir}, \mathsf{size}, \mathsf{lev}, \mathsf{slip})$ is submitted as

$$\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}_c}(i_t; r)$$

together with a zk-SNARK π_{valid} proving that ct encrypts a well-formed intent under a valid policy π, that the implied position remains within the user's risk caps, and that the user holds sufficient collateral. The matcher receives $(\mathsf{ct}, \pi_{\mathrm{valid}})$ and verifies π_{valid} without learning i_t.

5.3 Batch Matching

The matcher proceeds in discrete *epochs*, each of fixed wall-clock duration τ. Within an epoch, the matcher accumulates a multiset of valid encrypted intents

$$B = \{(\mathsf{ct}_j, \pi_{\mathrm{valid},j})\}_{j=1}^{m}.$$

At epoch close, the committee threshold-decrypts the intents and computes a fair clearing under the following rule.

Definition 5.1 (Epoch Clearing)**.** *Given a decrypted batch $\{i_j\}_{j=1}^m$, the epoch clearing is the assignment of fills $(f_j)_{j=1}^m$ that maximizes the volume crossed subject to (i) no fill exceeds its intent's size, (ii) all fills clear at a single uniform price p^* per instrument, and (iii) tie-breaking among partial fills is performed by a verifiable random function seeded by the epoch hash.*

The uniform-price epoch auction defeats sandwich attacks by construction: an adversary who learns intents only after the epoch closes cannot insert orders at advantageous relative positions, because there are no relative positions within an epoch.

Theorem 5.1 (MEV Resistance)**.** *Under the assumption that the threshold encryption scheme is IND-CPA secure and the committee contains at least $n - t + 1$ honest members, no probabilistic polynomial-time adversary $\mathcal{A}$ external to the committee can extract value through ordering or insertion attacks against intents within a single epoch. Formally, for the standard MEV game $\mathsf{Game}_{\mathsf{MEV}}$,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MEV}}(\lambda) \leq \mathsf{negl}(\lambda).$$

Proof sketch. The adversary observes only ciphertexts within the epoch. By IND-CPA security of the threshold scheme, the adversary's view is computationally indistinguishable from a view in which all intent plaintexts are replaced by zero strings of equal length. In that simulated view the adversary has no information on which to base an ordering attack, so any advantage in the real view must reduce to an advantage against IND-CPA security, which is negligible. The honesty bound on the committee prevents premature decryption. □

5.4 Settlement

The clearing produces a single aggregate state transition published to Base, accompanied by a zk-SNARK π_{clear} proving that the clearing satisfies Definition 5.1 relative to the committed batch. The contract verifies π_{clear} and updates the global position commitment tree, the funding accumulator, and the collateral vault.

6. Zero-Knowledge Copy-Trading

A strategy operator on GitSync may wish to attract followers without revealing the strategy. We give a construction that supports this.

6.1 The Profitability Statement

Let S denote a strategy operator's agent. Over a window $[t_0, t_1]$, S executes a sequence of intents producing realized profit and loss $\Pi_{[t_0,t_1]} \in \mathbb{Z}$. The operator wishes to publish a commitment to Π, together with a proof that Π was honestly computed from intents that were themselves valid under the matching rules of Section 5, without revealing any individual intent.

Define the statement

$$\Phi(\Pi, h_{t_0}, h_{t_1}) := \begin{cases} \exists\, \{i_t\}_{t\in[t_0,t_1]},\, \{f_t\}_{t\in[t_0,t_1]} : \\ \quad \text{each } i_t \text{ was a valid intent under policy } \pi_S \\ \quad \text{each } f_t \text{ is the clearing of } i_t \text{ under the epoch matcher} \\ \quad \Pi = \sum_t \mathrm{PnL}(f_t) \\ \quad h_{t_0}, h_{t_1} \text{ are the matcher's batch roots at } t_0, t_1. \end{cases}$$

6.2 Construction

The operator publishes a commitment $\bar{\Pi} = \mathsf{Commit}(\Pi; r)$ together with a zk-SNARK π_Π proving Φ. Followers verify π_Π in constant time using the verifying key on Base. The proof reveals nothing about the underlying intents.

Theorem 6.1 (Soundness and Privacy of Copy-Trading)**.** *Under the knowledge soundness of the proof system and the hiding property of* Commit*, the construction satisfies:*

1. ***Soundness.*** *No PPT operator can publish a proof π_Π that verifies for a false Π except with negligible probability.*
2. ***Privacy.*** *For any two strategy histories yielding the same Π and the same window roots, the operator's view emitted to followers is computationally indistinguishable.*

6.3 Follower Subscription and Royalty Flow

A follower opts in by depositing collateral and selecting a strategy operator. The follower's agent receives an additional input at each tick: a *shadow intent* produced by the operator's agent, encrypted under the follower's key. The follower's agent executes the shadow intent,

scaled to the follower's risk profile and subject to the follower's policy caps. A royalty equal to a configurable share of the follower's gross fees is paid to the operator, denominated in $GSYNC.

Remark 6.1. The follower does not see the operator's strategy. The follower sees only the agent's explanation e_t, which is generated independently for the follower's account from the executed intent. This preserves explainability without leaking strategy.

7. Threat Model and Security Claims

We state the threat model explicitly.

7.1 Adversaries Considered

$\mathcal{A}_{\textbf{searcher}}$. A computationally bounded adversary with read access to Base's public state, including all confirmed transactions, mempool observations, and oracle updates. Can submit transactions through any channel.

$\mathcal{A}_{\textbf{builder}}$. A computationally bounded adversary that, in addition to the searcher's capabilities, controls block construction for the Base sequencer for a bounded period.

$\mathcal{A}_{\textbf{committee-minority}}$. A computationally bounded adversary that corrupts up to $t-1$ members of the privacy committee.

$\mathcal{A}_{\textbf{counterparty}}$. A rational user who interacts with the protocol with the goal of extracting value from other users' visible positions.

7.2 Assumptions

Assumption 7.1 (Committee honesty)**.** The privacy committee is operated under a (t, n) threshold with $t > n/2$. At least $n - t + 1$ committee members are honest at all times.

Assumption 7.2 (Cryptographic hardness)**.** The threshold encryption scheme is IND-CPA secure, the zk-SNARK is knowledge sound and zero-knowledge, the commitment scheme is hiding and binding, and the hash function is collision resistant.

Assumption 7.3 (Base liveness)**.** Base provides finality within a bounded delay δ and does not censor transactions for periods exceeding δ_{escape}.

7.3 Security Claims

We claim, under the stated assumptions, the following.

Proposition 7.1 (Intent confidentiality)**.** *For any intent i_t submitted before its epoch closes, no adversary in $\mathcal{A}_{searcher} \cup \mathcal{A}_{builder} \cup \mathcal{A}_{counterparty} \cup \mathcal{A}_{committee\text{-}minority}$ can recover i_t except with probability negligible in λ.*

Proposition 7.2 (Position confidentiality)**.** *For any open position commitment C, no adversary in the above set can determine the position's direction, size, leverage, or liquidation price except with probability negligible in λ.*

Proposition 7.3 (Strategy confidentiality)**.** *For a strategy operator publishing copy-trading proofs as in Section 6, no follower or external observer can extract more information about the operator's strategy than is implied by the published profit-and-loss series.*

Proposition 7.4 (No MEV by external actors)**.** *Theorem 5.1 holds; external sandwich, front-run, and back-run attacks against epoch participants succeed with negligible probability.*

7.4 What GitSync Does Not Protect Against

In the interest of intellectual honesty we enumerate the residual surface.

- Aggregate inference. The aggregate clearing per epoch is public. Sophisticated observers may infer macro flow direction from aggregate statistics. Per-user attribution is not feasible, but population-level inference is.
- Committee collusion. If more than $t - 1$ committee members collude they can decrypt intents before clearing. The privacy committee is therefore composed under stake-weighted, slashable, geographically diversified operator selection (Section 8).
- Endpoint compromise. A compromised user device leaks the user's policy directly. Out of scope.
- Oracle manipulation. GitSync inherits its index price feed from established oracle infrastructure on Base. Manipulation of that feed is a shared risk across the venue.

8. Tokenomics

$GSYNCis the native token of the GitSync protocol. Its design follows three principles: maximal community alignment through a fair launch, fee accrual that is mechanical rather than discretionary, and operational sustainability funded by realized protocol revenue rather than by token emission.

8.1 Supply

The total fixed supply is 100,000,000 $GSYNC, distributed at genesis according to Table 1.

Allocation	**Share**
Public liquidity (fair launch)	96.0%
Marketing and development	4.0%
Total	**100.0%**

Table 1: $GSYNCinitial supply allocation.

The allocation reflects a deliberate choice: 96% of the total supply enters the open market at genesis, available to any participant on equal terms. No tokens are reserved for venture investors, no tranches are gated by private rounds, and no insider allocation accumulates beneath the surface of the launch. The remaining 4% funds the initial marketing and development effort required to bring the protocol from launch to self-sustaining revenue.

8.2 Operational Funding Model

Conventional protocol designs reserve a substantial fraction of supply for ongoing operations: treasury reserves, contributor compensation, staking incentives, ecosystem grants. GitSync rejects that pattern. After the 4% marketing and development allocation, all ongoing operations are funded by protocol revenue, denominated in collateral, accrued from realized trading activity. This has three consequences.

First, the protocol's solvency is tied directly to its usage. If GitSync produces no trading volume, no funding flows; if GitSync produces meaningful volume, the operational cost surface is covered without diluting holders. Second, contributors and operators are aligned with users

by construction: their compensation comes from the same fees users pay, not from a separate token supply they were granted. Third, the protocol's growth is constrained to what real usage can support, which we consider a feature rather than a limitation.

8.3 Revenue Streams

Protocol revenue accrues from four sources, each denominated in collateral (USDC).

1. **Trading fees.** A taker fee is applied to each filled intent. A fraction ρ_t of trading fees accrues to the protocol; the remainder rebates to passive liquidity providers in the collateral vault.
2. **Premium agent subscriptions.** Users may subscribe to higher-tier agent templates, denominated in $GSYNCpurchased on the open market. Subscription proceeds flow to the protocol.
3. **Agent marketplace fees.** The protocol collects a fee ρ_m on royalty flow between followers and template publishers.
4. **Copy-trading royalties.** The protocol collects a fee ρ_c on royalty flow between followers and strategy operators.

8.4 Fee Flow

Collected protocol revenue is allocated in the following order, with the proportions set by governance after Phase III.

1. **Privacy committee compensation.** Committee operators are paid for the epochs they participated in, proportional to stake and uptime. Compensation comes from realized fees, not from a reserved token pool.
2. **Open-market $GSYNCbuyback.** A fraction of revenue is used to buy $GSYNCon the open market. Bought tokens are distributed to reputation-stakers and committee-stakers in proportion to their stake.
3. **Insurance fund replenishment.** A fraction is directed to the insurance fund that covers under-collateralized liquidations.
4. **Ongoing development.** A fraction supports continued protocol development and audit cycles, paid in collateral to contributors.

This structure eliminates the need for any treasury or staking emission held as token supply: the productive role of $GSYNCis enforced by the buyback mechanic acting on the float, rather than by a separate reserved allocation.

8.5 Staking

$GSYNCmay be staked for two purposes. In both cases, the staked tokens are acquired on the open market; there is no protocol-issued stake.

Privacy committee staking. Staked $GSYNCqualifies an operator to participate in the threshold privacy committee. Operators earn a share of buyback distributions proportional to stake and uptime. Slashing applies for provable misbehavior: premature decryption, equivocation, or extended downtime.

Reputation staking. Agent template publishers and strategy operators stake $GSYNCagainst their published templates and strategies. Slashing applies for demonstrated policy violation.

9. Roadmap

GitSync deploys in four phases.

Phase I (Q2–Q3 2026): Confidential Intent Matcher. Initial deployment on Base mainnet. Encrypted intents, epoch matching, and public positions. AI agents in supervised mode with explanation gating. Target: closed beta to invited users.

Phase II (Q4 2026): Shielded Positions. Position commitments and nullifier set replace public position records. zk-SNARK circuits audited and deployed. Liquidations execute against commitments without revealing individual positions.

Phase III (Q1 2027): Copy-Trading and Agent Marketplace. Zero-knowledge copy-trading proofs go live. Agent template registry and royalty flow open to third-party publishers.

Phase IV (Q2 2027 and beyond): Cross-Venue Routing. Agents acquire the capability to source liquidity across multiple Base-native perpetual venues while preserving the confidentiality envelope. Bridging to Optimism Superchain neighbors under evaluation.

10. Related Work

GitSync sits at the intersection of three lines of work.

On-chain perpetuals. Perpetual Protocol, dYdX, GMX, Synthetix Perps, and Hyperliquid established the design space. Each publishes positions and intents in the clear and so faces the leakage problem of Section 2. Hyperliquid's order book is fastest in execution but is not confidential.

MEV resistance. CowSwap and 1inch Fusion introduced batch auctions for spot. Penumbra and Ferveo formalized threshold-encrypted mempool designs. GitSync applies the threshold-mempool primitive to perpetual futures specifically and extends it with the position-commitment layer.

Verifiable AI execution. EZKL, RISC Zero, and Modulus Labs provide zero-knowledge machine learning primitives. GitSync's copy-trading protocol uses standard zk-SNARK circuits rather than zkML proofs of inference, because the property being proved is profit-and-loss derivation, not model honesty. zkML may be adopted in a later phase for agent integrity guarantees.

11. Conclusion

GitSync proposes that the next epoch of on-chain derivatives will be defined by what the protocol does not show. Public order books, public positions, public intents, and public strategies were rational at the moment of bootstrapping decentralized derivatives, but they are no longer the frontier. They are the surface against which the next generation of extraction operates.

By relocating execution to private AI agents, by routing intent through a threshold-encrypted matcher, by representing positions as commitments, and by discharging copy-trading proof obligations to zero-knowledge primitives, GitSync removes that surface. The result is an exchange in which users delegate decisions to agents that work for them, with explanations rendered in plain language and trades executed without observers.

Trade with AI. Stay invisible.

References

[1] Hyperliquid Labs. *Hyperliquid: A High-Performance Decentralized Derivatives Exchange.* 2024.

[2] dYdX Trading Inc. *dYdX v4: Decentralized Perpetual Futures.* 2023.

[3] Y. Lin, A. Yang. *Penumbra: A Shielded, Cross-Chain Network.* 2022.

[4] J. Bonneau, J. Clark, S. Goldfeder. *On Bitcoin as a Public Randomness Source.* 2015.

[5] P. Daian et al. *Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges.* 2019.

[6] J. Groth. *On the Size of Pairing-Based Non-Interactive Arguments.* EUROCRYPT 2016.

[7] E. Ben-Sasson et al. *Scalable, Transparent, and Post-Quantum Secure Computational Integrity.* 2018.

[8] V. Buterin et al. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.* 2014.

[9] Coinbase. *Base: An Ethereum Layer 2 Network.* 2023.

[10] CowSwap. *Batch Auctions for Order Flow.* Technical documentation, 2022.

[11] Ferveo. *Threshold Encryption for Mempool Privacy.* Anoma research, 2022.

[12] EZKL. *Verified Inference for Neural Networks.* Technical documentation, 2024.

[13] Synthetix. *Synthetix V3 and Perps V2.* 2023.

[14] Perpetual Protocol. *The Virtual AMM and Funding Rate Mechanics.* 2020.

[15] GMX. *Decentralized Perpetual Exchange Architecture.* 2022.

This document is informational. It does not constitute an offer to sell or a solicitation of an offer to buy any security or financial instrument. Trading perpetual futures involves significant risk. Past performance does not guarantee future results. The $GSYNCtoken is a utility token of the GitSync protocol; it is not equity, and confers no rights to dividends or claims on assets.