WORKING PAPER

ClawFarm

A Protocol for Mining AI Inference

C. Wren

cwren.cf@protonmail.com

VERSION 2.1 · GENESIS DRAFT

Contents

ClawFarm: A Protocol for Mining AI Inference

C. Wren

cwren.cf@protonmail.com · Version 2.1 · Genesis Draft

ABSTRACT

We introduce ClawFarm, a protocol for the production and exchange of artificial intelligence inference, deployed as a set of immutabl

The token supply is allocated to two emission pools (Provider 70%, Developer 30%); emission follows a fixed schedule defined at Genesis. By contrast, is artificial intelligence inference, an economic output that the world consumes every day, in quantities that are growing at multiples per year.

The protocol treats inference capacity as fungible across three independent supply sources. Providers may resell closed-source API access obtained through one of the many discount tiers operated by frontier laboratories, capturing arbitrage spreads against list pricing. Providers may resell capacity obtained through consumer-tier subscriptions, capturing the subsidy implicit in those plans. Providers may operate their own infrastructure running open-weight models under permissive licenses, capturing the spread between their marginal cost and the price the market clears at. The protocol does not ask which of these three a provider draws from. It does not verify. It is, by design, structurally incapable of verifying. This neutrality is the protocol's primary mechanism for admitting capacity at scale.

The economic engine is the interaction between price-weighted CLAF emission and automatic buyback-and-burn of the three percent protocol fee. The emission's weighting term $P_{\text{avg}} / P_{\text{i}}$ creates the conditions under which providers may rationally operate below their immediate USDC marginal cost, financing the gap through accumulated CLAF rewards in expectation of the token's appreciation as protocol usage scales. This is the structural equivalent of Bitcoin's block reward subsidy of miners who operate above their immediate electricity cost. Combined with the deterministic buyback-and-burn of all collected fees, the mechanism produces a token whose circulating supply contracts monotonically in protocol usage — and an asymmetric pricing advantage against any centralized inference provider, whose marginal cost forms a floor that no protocol-external subsidy can credibly breach.

We argue that the construction described here cannot be implemented under any organizational form other than an immutable, pseudonymously-authored protocol with zero allocation. The protocol's load-bearing features: non-custodial settlement, structural neutrality across supply sources, fixed monetary schedule, and parameters defined at Genesis, collectively require credible permanence over the timescales relevant to participants. Pseudonymous authorship and zero allocation are structural choices that make the monetary mechanism legible from deployed rules rather than institutional promises.

preconditions under which the proposed cost-subsidy game can sustain itself.

This paper is intended as a technical specification sufficient for independent re-implementation, and as a statement of the protocol's economic design sufficient for independent evaluation. All on-chain logic is open source; this document describes its semantics. We adopt the convention that the protocol is referred to in the present tense as if already deployed; the reader should understand this as describing the system that the Genesis transaction will instantiate.

Note on this version

Version 1.0 of this paper, published in early 2026, described ClawFarm as a "settlement layer for AI agents." That framing was an understatement of the protocol's claim. The settlement function — non-custodial escrow, dual-signed metering, deterministic distribution — was correctly described, but it is not what the protocol does. It is what the protocol uses. The protocol is concerned with a more general question: under what conditions can an entire economic activity — the production and exchange of AI inference — be coordinated by an open protocol rather than by a set of firms.

We have rewritten the paper to make this claim directly. The substantive technical content of v1.0 is preserved in §5 through §7 and §10 through §13 with only minor corrections. The economic and monetary chapters (§8, §9, §12) have been substantially expanded. A new chapter (§4) describes the three-layer supply architecture that v1.0 implied but did not name. The introduction (§1) has been replaced.

Readers familiar with v1.0 may proceed directly to §4, §8, §9, and §12. Readers new to the protocol should read the paper in order. The technical sections (§5–§7, §10–§11) presume familiarity with the Solana program model; the economic sections (§4, §8, §9, §14) presume familiarity with first-year microeconomics and with Bitcoin's monetary design.

1. Introduction

1.1 Bitcoin's monetary innovation

Before discussing AI inference, this paper proposes a particular reading of what Bitcoin accomplished, because the same reading frames everything that follows.

The standard reading of Bitcoin emphasizes its monetary properties: a fixed supply schedule, the prevention of double-spending without a trusted intermediary, censorship resistance against state-level adversaries. These properties are real and consequential. But they are not, in our view, Bitcoin's primary innovation.

Bitcoin's primary innovation is that it demonstrated a category of economic system previously thought impossible: a permissionless market in which production, exchange, and monetary policy operate without any of the institutional preconditions that economic theory had assumed those functions required. Issuance, producer admission, participant coordination, and rule persistence are implemented through protocol mechanics rather than institutional discretion. The system runs because its participants find it rational to participate, and it continues to run through the incentives encoded in its rules.

What made this possible was a specific cryptoeconomic construction. Miners spend electricity on a computation whose only economic value comes from the protocol's commitment to reward valid blocks with newly issued bitcoin. The protocol, in turn, fixes a supply schedule that cannot be modified. The result is a self-referential equilibrium: miners accept short-term electricity losses because they expect the protocol's token to appreciate; the token appreciates because the protocol successfully constrains its own supply; and the protocol successfully constrains its supply because miners enforce the constraint by mining. The system has, for sixteen years, been impossible to break and impossible to dismantle.

We propose that an analogous construction is now possible for the production and exchange of AI inference, and that the structural conditions that allowed Bitcoin to succeed are required, with no available substitution, for any protocol that attempts it.

1.2 Inference enters its monetary moment

The conditions under which a Bitcoin-like construction becomes possible for any given economic activity are, in our reading, three.

First, the underlying production technology must be commoditized. Anyone willing to acquire the relevant equipment must be able to perform the productive activity without permission. For monetary production, this condition was met when general-purpose computation became cheap enough that hashing became a viable amateur activity. For inference production, this condition is met in 2026: open-weight language models of frontier-adjacent quality — DeepSeek V4 under MIT, Kimi K2.5 under MIT, Qwen 3.5 under Apache 2.0, GLM-5 under MIT, Llama 4 under Meta's permissive license — are publicly available, and the GPU hardware required to serve them spans from consumer-grade RTX 5090 cards capable of serving distilled models, to small clusters of four to eight H100s capable of serving frontier mixture-of-experts models. The production technology is no longer gatekept.

Second, a settlement layer must exist that can record transactions in a form acceptable to producers and consumers without intermediary trust. For Bitcoin, this was Bitcoin itself, bootstrapping. For inference, this is the existence of a high-throughput blockchain capable of settling thousands of transactions per second with sub-second confirmation, paired with a stable medium of exchange that producers can hold without intermediary risk. As of 2026, this condition is met by the combination of Solana's L1 throughput and the regulatory acceptance of USDC as a transactable stablecoin.

Third, there must be a sufficient cost asymmetry between the existing incumbent producers and the prospective protocol-mediated producers that, even after accounting for the protocol's overhead, the new market can offer a meaningfully better deal to one or both sides. For Bitcoin, the asymmetry was that central banks could not match Bitcoin's credible commitment to a fixed supply at any cost. For inference, the asymmetry is multi-layered.

The frontier laboratories operating closed-source inference services have, as of mid-2026, gross margins between forty and sixty percent on inference revenue. These margins fund research, brand, sales operations, and substantial subsidies to retain consumer subscribers. The aggregators sitting on top of these laboratories add another five to fifteen percent in their own margin. Self-hosted open-weight inference, conducted on commodity GPU infrastructure, runs at a marginal cost that is, for comparable-quality output on non-frontier tasks, between five and fifty times lower than the closed-source list price. A protocol that does not need to fund research, brand, sales, or subscriber subsidies can, in principle, pass the entire spread to its participants — provided it has a mechanism to bootstrap liquidity without itself needing to fund operations.

This paper describes such a mechanism. The remainder of this introduction frames it.

1.3 The three sources of inference supply

The protocol admits inference capacity from three independent sources, which we treat throughout this paper as fungible. We name them here and develop them in §4.

Layer A — Closed-source resale. Providers redistribute capacity obtained through accounts they hold with frontier laboratories, capturing the spread between the discount tier they themselves access and the prices at which the protocol's market clears.

Layer B — Subscription credit secondary market. Providers redistribute capacity obtained through consumer-tier subscriptions whose implicit subsidy substantially exceeds the subscription price.

Layer C — Open-weight self-hosted inference. Providers operate GPU infrastructure under their own control, running open-weight language models obtained under permissive licenses, at marginal costs determined by hardware amortization and electricity.

The protocol does not distinguish among these sources. A user submitting a request is matched, through the routing logic described in §10, to a provider declared to serve the requested model at acceptable price and quality. The protocol records what was served, what was paid, and to whom. It does not record, and cannot verify, what underlay the provider's ability to serve.

This neutrality is not a regulatory posture. It is the protocol's primary mechanism for admitting capacity at scale.

1.4 Contributions

The contributions of this paper are as follows.

First, we present a protocol for the production and exchange of AI inference whose design lineage is Bitcoin, in a sense defined by permissionless production, fixed monetary schedule, and parameters defined and persistent for the emission lifecycle. The construction is specified in sufficient detail to be reimplemented by a competent reader given access to the Solana program model.

Second, we describe the three-layer supply architecture by which the protocol admits capacity from closed-source resale, subscription credit secondary markets, and open-weight self-hosted inference simultaneously, and we argue that the protocol's structural inability to distinguish among these layers is a feature without which the construction cannot scale.

Third, we describe the price-relative emission mechanism by which the protocol's reward token, CLAF, subsidizes provider operation below immediate USDC marginal cost. We argue that this mechanism is the structural equivalent of Bitcoin's block reward subsidy of mining electricity, and that it produces a pricing equilibrium that no centralized inference provider can replicate.

Fourth, we describe an automated buyback-and-burn mechanism for the three percent protocol fee, implemented through a cross-program invocation to the Jupiter aggregator on Solana. Conditions are evaluated by the contract at epoch boundaries; execution is automatic when thresholds are met, subject to liquidity-bounded slippage constraints.

Fifth, we argue (in §12, recognizing that the argument is unusual) that the load-bearing features of the protocol require operations that are fully on-chain after Genesis. We discuss this with reference to the Bitcoin precedent and with reference to several attempted alternatives.

1.5 What this paper is not

This paper does not propose a new approach to AI itself. ClawFarm does not improve the quality of any model, reduce the cost of any individual inference operation, or change the latency at which any provider can serve a request. The technical layer at which we operate is strictly below these concerns: we are concerned only with the financial and informational coordination that mediates between a user and a model, regardless of which model, regardless of who hosts it. A reader expecting a contribution to the science of language models, multimodal reasoning, or agent architecture will not find one here.

Nor is this paper an investment thesis. CLAF, the protocol's reward token, neither represents equity nor confers governance rights. It functions as the reward unit for inference settlement. Its distribution tracks verified activity on the protocol, in fixed proportions defined at Genesis. We discuss its emission schedule in §8 because the schedule is consequential to the protocol's incentive structure, not because the schedule constitutes a value proposition.

The emission pools are available to participants according to the same reward formulas, through verified protocol activity.

Finally, this paper does not claim that the construction it describes is the only possible design for a permissionless AI inference market. We claim only that it is one design, that the design is internally consistent, that its mechanisms admit the analyses we present, and that the conditions under which it can be deployed are present in the AI inference market as of mid-2026. The construction may succeed or fail. If it fails, it should be possible to determine, from this document and from the on-chain history that will accumulate after Genesis, why.

⁂

2. Background and Related Work

The protocol draws from three traditions in distributed systems research: Bitcoin and the broader literature on cryptoeconomic protocols; decentralized physical infrastructure networks (DePIN) for the coordination of physical capacity; and centralized inference aggregation as the current dominant solution to the AI inference market's coordination problem. We review each, with an emphasis on what the protocol takes from it and what it does not.

2.1 Bitcoin as design lineage

We treat Bitcoin not primarily as a payment system, nor as a monetary alternative to fiat currency, but as a class of cryptoeconomic construction with specific structural properties. The properties we inherit, in approximately their original form, are five.

Permissionless production. Any party with the relevant equipment may produce the protocol's commodity (in Bitcoin: valid blocks; in ClawFarm: inference responses) and receive the protocol's reward. Producer admission is governed by registration requirements rather than by an operator. Bitcoin's mining requires only a node and hashing capacity; ClawFarm's provider registration requires only a Solana wallet, the posting of a 100 USDC bond, and the technical capacity to serve inference requests under the protocol's metering scheme.

Fixed monetary schedule. The total supply of the protocol's reward token is fixed at Genesis. The rate at which the supply is emitted follows a publicly known schedule whose parameters cannot be modified. Bitcoin's schedule halves every 210,000 blocks, approximately every four years; ClawFarm's halves every two years, with five halving periods over a ten-year emission horizon.

Token supply. The full 1B CLAF is allocated to two emission pools at Genesis (Provider Pool 70%, Developer Pool 30%), distributed over 10 years per the fixed schedule. The first CLAF in circulation will be the first CLAF emitted by the protocol to a verified participant.

Upgrade authority. Solana program upgrade authority is renounced at Genesis; the rules are set at deployment and persist for the protocol's emission lifecycle. Bitcoin's modifications require coordinated adoption by miners and node operators; ClawFarm fixes its program parameters by deploying its programs with renounced upgrade authority. Once Genesis is executed, the parameters are mathematically immutable.

Pseudonymous authorship. The protocol is authored under a pseudonym. The author neither holds tokens beyond what is earned through participation, nor retains operational control over the protocol after deployment, nor speaks publicly to influence the market for the protocol's token. Bitcoin's case is the canonical one: Satoshi Nakamoto's anonymity has, in our reading, been a structural contributor to Bitcoin's credibility, not a contingent biographical fact. ClawFarm adopts the same convention for the reasons set out in §12.

What we do not inherit from Bitcoin is Bitcoin's specific consensus mechanism, its specific block structure, or its specific approach to settlement finality. ClawFarm runs on Solana; consensus and finality are properties of the underlying chain. The protocol contributes nothing to consensus research and assumes the underlying chain operates correctly. We discuss the implications of this dependence in §11.5.

2.2 Why prior decentralized compute protocols are not Bitcoin for inference

Several existing protocols in the decentralized physical infrastructure category — Bittensor, Render, Akash, Filecoin, Helium, and others — coordinate physical capacity through token incentives. We discuss each briefly, with attention to the dimensions on which ClawFarm differs.

Bittensor coordinates a network of "subnets," each of which is a specialized market for some AI service, with rewards distributed through a yuma-consensus-based validation mechanism. Bittensor's market structure is hierarchical: subnet operators control validator admission, validators score miners, and the Open-Tensor Foundation retains substantial governance authority over the protocol. The April 2026 exit of Covenant AI, accompanied by allegations of centralized control by the protocol's co-founder, illustrates the consequences of this structure. ClawFarm does not use subnets or Bittensor-style validators. Provider scoring is automated through the dual-signed metering scheme described in §6, not through validator consensus.

Render coordinates GPU capacity for graphics rendering and, more recently, AI inference, with a burn-and-mint equilibrium token model in which network usage drives RENDER token burns. Render operates as a registered company (OTOY) with foundation governance, with a substantial allocation to the team and to early investors, and with operational marketing apparatus. The protocol's pricing and provider selection are operated by the company. ClawFarm has no equivalent operational layer.

Akash coordinates general-purpose compute capacity, including GPU compute, through a bid-and-ask mechanism with on-chain settlement. Akash is closer to ClawFarm in spirit than Bittensor or Render, but its scope is broader (general compute) and its mechanism is less specialized for the particular structure of AI inference markets — in particular, Akash does not provide the cost-subsidy game described in §9, which we believe is necessary to bootstrap an inference market against well-funded centralized competitors.

Filecoin coordinates decentralized storage with proof-of-spacetime, providing a useful structural reference for the verification of physical capacity. ClawFarm's metering scheme is not analogous — inference cannot be verified through a periodic challenge in the way storage can — but Filecoin's experience with the divergence between physical capacity and productive use informs our treatment of the quality factor Q in §6.

The common feature distinguishing ClawFarm from each of the above is that, in each case, there is a known organizational entity behind the protocol, holding tokens, retaining governance authority, and operating a marketing apparatus to drive adoption. In each case, the entity is, in our reading, a structural impediment to the protocol's credible commitment to its own rules. We will return to this point in §12.

2.3 Why centralized aggregators cannot occupy this design space

The current dominant solution to the AI inference market's coordination problem is the centralized aggregator. OpenRouter, Together AI, Fireworks, Replicate, Anyscale, and dozens of others perform routing, metering, and settlement between many providers and many users. Aggregators have proven the demand for the function: OpenRouter's annualized inference spend reached one hundred million dollars by mid-2025 and approximately five hundred million by mid-2026, with the platform charging approximately five percent on top of the inference cost.

The question this paper takes seriously is whether an aggregator could, with sufficient willingness, evolve into the construction we describe. We argue that it could not, for three reasons.

First, an aggregator's pricing power derives from its position as the intermediary that selects providers and meters consumption. The five percent margin OpenRouter charges is the rent it extracts for performing those functions. An aggregator that voluntarily reduced this rent to three percent — and committed to remitting all of it to a buyback-and-burn — would be giving up its principal source of revenue. An equity-financed firm cannot credibly make such a commitment in perpetuity, because its equity owners retain the option to reverse the commitment at any time. The commitment is, in the contractual sense in which it would have to

be made, not credible.

Second, an aggregator's value to providers depends on its ability to verify provider quality and to remit payments reliably. To perform these functions, the aggregator necessarily maintains records of provider identities, transaction histories, and operational practices. These records create a disclosure surface that exposes the aggregator to legal action by upstream parties whose terms of service its providers may violate. An aggregator that admitted Layer B providers (consumer subscription resellers) at scale would expose itself to action by the consumer subscription issuers. The aggregator's structural ability to identify Layer B

activity is precisely what allows it to be ordered to disclose that activity. ClawFarm, by contrast, omits supply-source records by design and provides no mechanism to acquire them.

Third, an aggregator cannot run the cost-subsidy game described in §9. An aggregator's revenue is, by accounting necessity, denominated in fiat or stablecoin. It cannot subsidize providers below their marginal cost without absorbing the loss itself, and it has no mechanism to recover the loss through token appreciation, because it does not issue a token. Even if an aggregator attempted to issue a token, the token would carry the aggregator's contingent liabilities as overhang, and the market would correctly discount it. The cost-subsidy game is, structurally, the property of protocols rather than firms.

We do not claim that aggregators are inferior to ClawFarm on every dimension. For users requiring strong service-level agreements, regulated counterparties, fiat billing, or invoiced contracts, aggregators will continue to be the appropriate choice. We claim only that the specific construction described in this paper — permissionless production, credible monetary commitment, structural neutrality across supply sources, cost-subsidy bootstrap — is not available to any firm, by virtue of what a firm structurally is.

2.4 What ClawFarm is

For the avoidance of doubt, we state what ClawFarm is and what it is not.

ClawFarm is a set of four Solana programs — Provider Registry, Settlement Escrow, Token Metering, and Treasury — together with an SPL token, CLAF, and an off-chain routing layer that any participant may operate. The four programs are deployed once, at Genesis, and their upgrade authority is renounced in the same transaction. CLAF is minted to a treasury PDA in accordance with the emission schedule and distributed to verified participants in the proportions specified in §8.

ClawFarm is a set of deployed programs rather than a company. Protocol ownership, revenue custody, developer employment, and contract execution are outside the program interface. The authors of this paper are authors in the sense that an author writes a paper rather than the sense that a founder founds a firm. The protocol, once deployed, operates without their participation, and would continue to operate if they ceased to exist.

pl
a
se
fi
c

ClawFarm is public infrastructure: a set of programs deployed on a public blockchain, available for any party to use on terms encoded in the programs themselves and verifiable by inspection. Future modifications to the design, if any are wanted, would be implemented as separate protocols by separate authors, deploying separate programs. CLAF is a reward token. Settlement, emission, and buyback execute as protocol operations encoded at Genesis. The protocol's parameters are fixed at Genesis and cannot be modified by any aggregation of participants, however large.

⁂

3. System Model and Assumptions

We specify the threat and trust model under which the protocol is analyzed, the participants and their roles, and the underlying chain's properties on which the construction depends.

3.1 Participants

The protocol has four classes of participants.

Users are parties wishing to consume inference. A user maintains a Solana wallet, deposits USDC into a protocol-controlled escrow PDA, signs inference requests with their wallet's private key, and receives served responses from providers. Users are pseudonymous in the sense that they are identified only by Solana addresses; the protocol does not perform identity verification.

Providers are parties wishing to produce inference. A provider maintains a Solana wallet, posts a 100 USDC bond into a protocol-controlled bond PDA, registers in the Provider Registry program with the models and prices they offer, serves requests routed to them, co-signs usage proofs with users, and receives settlement in USDC and reward emission in CLAF. Providers are likewise pseudonymous.

Application operators are a subclass of users who serve inference to downstream end-users via their own interfaces. They maintain wallets identified to the protocol as application accounts, which qualify them for the demand-side reward emission share described in §8.

Challengers are any party — typically other providers, or motivated third parties — who post small bonds to challenge suspect settlements and receive a portion of slashed funds if challenges are upheld. Challengers play the role of a permissionless audit mechanism.

3.2 Trust assumptions

We make the following minimal trust assumptions.

We assume that the underlying Solana network operates correctly: that committed transactions are final, that program state is consistent across honest validators, that the network can sustain its stated throughput, and that no adversary controls a sufficient share of stake to revert committed history within the protocol's epoch boundaries. We acknowledge that this is a non-trivial assumption and discuss its limitations in §11.5.

We assume that cryptographic primitives — ed25519 signatures, sha-256 hashes, the Jupiter aggregator's price feeds — operate correctly under their standard security parameters.

We assume that USDC, the stablecoin in which all settlement is denominated, maintains its peg within tolerances that do not destabilize the protocol's economic mechanisms. The protocol's automated buyback-and-burn relies on USDC's continued tradability against CLAF on liquid markets; if USDC's peg breaks materially, the protocol's monetary mechanism is compromised. We treat this as an external risk and do not attempt to design around it.

We make no trust assumptions about providers, users, application operators, or challengers individually. Any of them may behave adversarially, may collude, and may attempt to extract value from the protocol through any available mechanism. The protocol's security properties (§11) are stated under these adversarial assumptions.

3.3 Threat model

We consider the following adversarial behaviors.

Provider misreporting. A provider may attempt to over-report token counts to inflate settlement amounts, or to under-deliver requested computation while claiming full delivery. The dual-signed metering scheme of §6 and the sampling audit of §11.3 are designed to bound the gains from such misreporting.

User repudiation. A user may attempt to repudiate service they in fact received, refusing to co-sign valid usage proofs. The escrow-and-timeout mechanism of §7.3 bounds the protocol's exposure to repudiation.

Sybil registration. An adversary may register many provider identities with bonded capital, attempting to capture disproportionate emission share. The nonlinearity of the reward function in price and quality (§8.3) makes Sybil registration unprofitable in expectation, and the 100 USDC bond imposes a per-identity capital cost.

Routing manipulation. Any off-chain routing service may attempt to bias request flow to favored providers. The protocol does not depend on any specific routing service; users may route directly, and competing routers create market discipline. We discuss this in §10.

Model identity fraud. A provider may register as serving one model and actually serve a different, cheaper, lower-quality model. The protocol cannot verify model identity in the general case; the quality factor Q (§6.4) and challenger-driven audit (§11.3) provide partial mitigation but do not eliminate this risk. We acknowledge the limitation in §15.

Collusion at small scale. If the provider population is small, collusion among providers to fix prices is possible. The price-relative emission term (§8.3) makes collusion against price reduction self-defeating in equilibrium, but small-population effects may exist at protocol bootstrap. We discuss this in §11.4.

Each of these threats is addressed in §11; we summarize them here only to clarify the scope of the analysis.

3.4 The underlying chain

The protocol is deployed on Solana for reasons we summarize here and do not return to.

Solana provides sub-second transaction finality, an SPL token standard suitable for high-throughput value transfer, a program model (Anchor / native Solana programs) suitable for the construction's complexity, native composability with

Jupiter aggregator for the buyback-and-burn mechanism, and an established market for SOL-denominated transaction fees that the protocol can absorb without imposing them on participants. The choice of Solana is not, in our reading, a claim that Solana is the optimal chain for protocols of this kind; it is a claim that Solana is a sufficient chain in 2026, that no alternative chain offered a meaningfully better tradeoff at the time of design, and that the choice forecloses certain future modifications (e.g., portability to other chains) that would in any case be prevented by the protocol's immutability.

The protocol does not depend on Solana's specific consensus mechanism or its specific approach to validator selection. If Solana's properties degrade — through validator capture, throughput collapse, or any other failure mode — the protocol's properties degrade correspondingly. We treat this as an external risk and do not design around it. A successor protocol on a different chain would, if implemented, be a different protocol; ClawFarm is, by construction, Solana-bound.

⁂

4. Three-Layer Supply Architecture

The protocol treats provider supply as fungible across three independent sources. We describe each, then argue that the protocol's structural neutrality across them is the central mechanism by which the construction admits capacity at scale.

4.1 Layer A — Closed-source API resale

A provider in Layer A holds an account, in their own name, with one or more closed-source inference services — OpenAI, Anthropic, Google, xAI, or others. They serve requests by making API calls to those services on behalf of ClawFarm users, paying the underlying service from their own balance and receiving USDC settlement through the protocol. From the protocol's perspective, the provider is a black box: they declare a model, they serve requests, and they sign usage proofs.

The economic basis for Layer A is the multi-tier discount structure that all major closed-source providers operate. We describe the structure briefly because its details are consequential to the arbitrage that providers in Layer A capture.

Frontier laboratories publish a list price. They also operate, in parallel, a set of discount tiers that are negotiated rather than published. A startup admitted to OpenAI's or Anthropic's startup program receives credits at substantial markdown to list — frequently fully subsidized for the first one to twenty-five thousand dollars of consumption. A volume-commitment customer receives twenty to forty percent off list. A strategic enterprise customer with annual commitments in the seven figures receives fifty to seventy percent off. A frontier-lab partner — typically a major model integrator, hardware partner, or co-marketing arrangement — receives seventy to ninety percent off. The variance across these tiers against a single "list price" that few customers actually pay creates a spread of three-to-one between the highest-discount and lowest-discount tiers, all on identical inference. A provider in Layer A captures this spread by serving the protocol's market at a price that lies between their access cost and the list price.

In addition to the discount-tier spread, two further arbitrages exist within Layer A.

Temporal arbitrage. All major closed-source services issue credits that expire twelve months after issuance. The marginal value of credits with thirty days of remaining life is, to their holder, zero. A secondary market for expiring credits has emerged independently of the protocol — the brokerage aicredits.co reports having transacted over twenty million dollars in such credits as of early 2026, with discounts of forty to sixty percent against face value and a ninety percent customer repeat rate. The protocol mechanizes this arbitrage on-chain, eliminating the broker overhead and enabling continuous price discovery rather than per-deal negotiation. Reported industry figures suggest that approximately five hundred million dollars of issued AI inference credits expire unused annually; this is the scale of the temporal-arbitrage opportunity that the protocol's Layer A admits.

Geographic arbitrage. Several closed-source services do not extend API access to a subset of jurisdictions (typically due to sanctions, regulatory uncertainty, or commercial choice). Inference demand in those jurisdictions is, on aggregate, substantial. A provider operating from a permitted jurisdiction may serve users in restricted ones through the protocol, capturing the geographic spread. Settlement in USDC, rather than fiat, removes the banking-infrastructure barrier that would otherwise prevent such cross-jurisdictional flows.

The combination of these three arbitrages — discount tier, temporal, geographic — defines the addressable market for Layer A. Within Layer A, the protocol's value proposition relative to centralized aggregators is twofold: structurally lower

take rate (three percent versus five to ten percent for most aggregators), and structural neutrality regarding provider identity (no know-your-customer beyond a Solana wallet).

4.2 Layer B — Subscription credit secondary market

A provider in Layer B holds consumer-tier subscriptions to closed-source inference services — typically twenty to two hundred dollars monthly, often bundled with desktop or mobile applications, and structured by the issuing laboratory to bundle substantially more inference value than the subscription price nominally pays for. The subsidy implicit in these plans is, as of mid-2026, often a factor of ten to twelve between price paid and inference value extractable through programmatic access. A provider in Layer B routes the inference flow through the protocol, capturing this subsidy gap.

We acknowledge, candidly, that Layer B is the most fragile of the three supply sources. The terms of service of consumer-tier plans typically prohibit programmatic resale, and the issuing laboratories have, throughout 2026, actively closed avenues by which subscription value can be extracted at scale. The April 2026 termination of OpenClaw's Pro-subscription routing — accompanied by the disclosure that OpenClaw users had been extracting approximately two hundred thirty-six dollars of API-equivalent value per twenty-dollar monthly subscription — and the May 2026 separation of Claude Code billing from seat-level subscriptions, with associated price adjustments of approximately ten to twelve times for non-interactive use, are recent examples. The economic window for Layer B is, in our reading, six to eighteen months in width before laboratory countermeasures substantially reduce its spread.

The protocol's neutrality across supply sources is, however, particularly consequential for Layer B. A provider operating in Layer B faces, at minimum, contractual exposure to the issuing laboratory. The protocol's inability to identify Layer B activity protects providers from the indirect channel by which laboratory countermeasures would otherwise reach them — namely, a subpoena to the protocol records linking Layer B participants to the protocol. In ClawFarm, supply-source data is not written on-chain, so disclosure through protocol records cannot occur.

We discuss the broader implications of this property in §12.4.

4.3 Layer C — Open-weight self-hosted inference

A provider in Layer C operates GPU infrastructure under their own control, running open-weight language models obtained under permissive licenses. Their marginal cost is hardware amortization, electricity, and operational overhead. Their price floor is determined by these real-world costs, not by any third party's terms or pricing decisions.

Layer C is, in our reading, the protocol's most consequential supply source. It is the source for which the Bitcoin analogy holds most directly. As Bitcoin miners produce monetary base from electricity and silicon, Layer C providers produce inference tokens from electricity and silicon, and exchange the result for USDC at protocol-mediated market prices. Layer C is not subject to any laboratory's terms of service. Its capacity cannot be revoked. It scales as global GPU inventory scales.

The economic conditions for Layer C are favorable as of mid-2026 and are improving. Open-weight inference benchmarks now approach closed-source inference quality for most non-frontier tasks. DeepSeek V4 Flash serves at a marginal cost of approximately twenty-eight cents per million tokens on a single H100, against closed-source list prices of one dollar fifty to two dollars fifty per million tokens for comparable-quality output. Llama 4 Maverick serves at approximately nineteen cents per million tokens in distributed deployment. Qwen 3.5 at the 235B parameter scale runs on commodity multi-GPU configurations costing approximately three thousand five hundred dollars in consumer hardware. GLM-5 at 744B parameters under MIT license, Kimi K2.5 at one trillion parameters under MIT license, and the broader open-weight model ecosystem provide a base of permissively-licensed model weights covering virtually every inference task at quality approaching the closed-source frontier.

The cost asymmetry between Layer C providers and centralized closed-source providers is, structurally, between five and fifty times in Layer C's favor for non-frontier tasks. This asymmetry is widening. The trajectory of open-weight model release cadence, the trajectory of GPU price-performance, and the trajectory of inference framework optimization (vLLM, SGLang, TensorRT-LLM) all point toward Layer C's advantage growing rather than shrinking.

We note explicitly that Layer C is the supply source whose economic conditions most directly determine the protocol's long-term viability. Layers A and B exist; they will provide capacity at scale during the protocol's early operation. But Layer A is fundamentally a redistribution of frontier-laboratory inference, and Layer

B is fundamentally an arbitrage of laboratory subsidies. Neither is a production-side innovation. Layer C is. The Bitcoin analogy, applied with care, predicts that Layer C will, over the protocol's emission horizon, dominate.

4.4 The protocol's structural neutrality

The protocol does not distinguish among providers by supply source. It does not ask which source a provider draws from. It does not verify, and it is technically incapable of verifying, the answer. We state this property formally and discuss its consequences.

From the protocol's perspective, a provider is a Solana wallet that has posted a 100 USDC bond into the provider bond PDA, registered in the Provider Registry program with one or more (model, price, quality) triples, and successfully co-signs usage proofs for requests routed to them. The protocol's state about a provider is precisely this: their address, their bond, their declared offerings, their accumulated served volume by epoch, and their resulting CLAF rewards. The protocol does not record which closed-source account, if any, the provider uses; which subscription, if any, the provider holds; which open-weight model, if any, the provider runs locally. The protocol interface omits this information, and on-chain state is not allocated to store it.

This property has two consequences worth stating directly.

The first consequence is that providers in Layer B, who would face laboratory action if their activity were attributable to a known intermediary, can participate in the protocol without exposing the intermediary to that action. The protocol cannot be sued by a laboratory to disclose Layer B providers, because the protocol does not know which providers are in Layer B. This is not a loophole. It is a direct consequence of the protocol's information architecture: a system that does not record the source of its capacity cannot be ordered to disclose it.

The second consequence is that Layer A, B, and C providers compete against each other on price within a single market. A Layer C provider serving open-weight inference at thirty cents per million tokens competes directly against a Layer A provider serving closed-source inference at the same price. Users select among them based on declared model, declared quality, and observable latency. The protocol's routing logic (§10) and verification scheme (§6) do not assume the existence of any specific supply source; they operate identically on all three.

The combination of these two consequences produces an outcome that no centralized aggregator can replicate: a single market for inference in which capacity from every available source is admitted without permission, and in which the protocol's structural inability to verify source is the precondition for capacity at scale.

4.5 Emission Pools

The protocol's reward token (CLAF) is allocated to two pools at Genesis, each receiving a fixed share of each epoch's emission:

Provider Pool (70%). Distributed to provider wallets that serve inference and settle USDC. Reward weight per provider is proportional to USDC revenue served in the epoch, with a price-relative modifier; providers priced below network average earn higher weight.

Developer Pool (30%). Distributed to wallets that call the protocol's API and pay USDC for inference. Reward weight per developer is proportional to USDC spent in the epoch. This pool exists to bootstrap demand-side participation: developers who consume the protocol accrue CLAF as an operational byproduct, not as a separate financial allocation.

Both pools follow the same emission schedule: 10-year emission curve, halving every 2 years, vested over 180 days linear.

5. Protocol Architecture

The protocol is implemented as four Solana programs operating in coordination. We describe each program's responsibilities, the on-chain accounts it manages, and the interactions among them. The pseudocode and account schemas are given in Appendices B and D.

5.1 Program 1: Provider Registry

The Provider Registry is responsible for the admission of providers, the management of their bonds, and the maintenance of their declared service offerings. The program exposes the following instructions: register_provider, which creates a new ProviderState account, accepts a 100 USDC bond deposit into the program's bond PDA, and emits a registration event; update_offering, which modifies a provider's declared (model, price, quality) triples; withdraw_bond, which permits a provider to retrieve their bond after a seven-day cooldown during which no challenges are outstanding; and slash_bond, callable only by the Settlement Escrow program in the course of a successful challenge.

The provider's bond is the protocol's mechanism for imposing a finite economic cost on adversarial registration. The 100 USDC quantum is chosen to be small enough to admit individual operators with consumer-grade hardware (a single RTX 5090, an H100 rental for a few weeks), and large enough that a one-time Sybil attack to capture an epoch's emissions would require capital outlay proportional to the attack's expected return. We discuss the bond's sufficiency under various attack models in §11.2.

5.2 Program 2: Settlement Escrow

The Settlement Escrow program manages user funds, processes settlements, and handles challenges. The program exposes: `deposit`, which moves USDC from a user's wallet into a per-user escrow PDA; `open_session`, which reserves a session-level allocation against a user's escrow balance for a particular provider; `settle_session`, which accepts a dual-signed usage proof, deducts the corresponding USDC from the user's escrow, transfers ninety-seven percent to the provider, and transfers three percent to the Treasury program's collection PDA; `challenge_settlement`, which permits any party to post a 2 USDC challenge bond against a specific settlement, triggering a challenge window during which the challenged provider must produce evidence; and `resolve_challenge`, which adjudicates the challenge and slashes or releases bonds as appropriate.

User funds are held in PDAs whose seed is derived from the user's wallet address; the program-derived address (PDA) is controlled by the Settlement Escrow program itself, not by any external key. Funds move out of an escrow PDA only through the program's instructions, which require either a user signature (for deposit, withdrawal, session opening) or a valid dual-signed proof (for settlement). This property is non-trivial: a private key controlling a PDA's authority cannot exist, by construction of the PDA derivation mechanism. The protocol's non-custodial property follows from this fact and is verifiable on-chain.

5.3 Program 3: Token Metering

The Token Metering program is responsible for the verification of usage proofs and the aggregation of per-epoch provider statistics that determine reward emission. Instructions include `submit_proof`, which validates that a settlement's accompanying usage proof carries valid user and provider signatures and that its claimed token counts fall within bounds derived from the model's declared tokenization; `aggregate_epoch`, which finalizes per-provider epoch statistics (total tokens served, volume-weighted average price, quality factor) at each epoch boundary; and `compute_reward_weight`, which applies the reward weight function of §8.3 to produce per-provider CLAF emission claims.

The metering program does not store individual proofs after they have been validated; only the aggregated per-epoch statistics persist. This minimizes the program's on-chain storage requirements and bounds the protocol's data exposure.

5.4 Program 4: Treasury

The Treasury program holds the protocol's accumulated fee revenue, executes the automated buyback-and-burn, and manages CLAF emission. Instructions: `collect_fees`, callable by Settlement Escrow upon settlement, which transfers three percent USDC into the treasury collection PDA; `execute_buyback`, callable by any party as a permissionless cron-like trigger at each epoch boundary, which invokes the Jupiter aggregator to swap accumulated USDC for CLAF under specified slippage and volume constraints, then transfers the acquired CLAF to the burn address; `mint_emission`, callable at each epoch boundary, which mints the epoch's CLAF emission to a distribution PDA; and `claim_reward`, callable by any verified participant, which transfers their epoch's claimed CLAF (computed by Token Metering) from the distribution PDA to their wallet.

> The Treasury program executes as a deterministic state machine. Its operations are fully deterministic: given the epoch's settled volume, the buyback amount is fixed; given the epoch's verified participants, the emission distribution is fixed; the burn address is a constant.

o
a
is

5.5 Inter-program data flow

The end-to-end data flow for a single inference transaction is the following.

A user, having previously deposited USDC into their escrow PDA via Settlement Escrow's `deposit` instruction, signs an inference request and transmits it to a chosen provider. The provider, having previously registered via Provider Registry's `register_provider` and posted their bond, receives the request, serves it, computes the resulting token counts, and returns the response together with a usage proof signed by the provider. The user verifies the response, co-signs the usage proof, and submits the dual-signed proof to Settlement Escrow's `settle_session` instruction. Settlement Escrow validates the proof's signatures (by calling into Token Metering's `submit_proof`), computes the USDC amount due, deducts that amount from the user's escrow PDA, transfers ninety-seven percent to the provider's wallet, and transfers three percent to the Treasury's collection PDA.

In addition to USDC settlement, each settlement event triggers CLAF emission events to both Provider Pool and Developer Pool, weighted by the metrics defined in §4.5.

At each epoch boundary (we set the epoch length at one hour for emission distribution, with longer rollups for halving periods), Token Metering's `aggregate_epoch` finalizes per-provider statistics. Treasury's `execute_buyback` may be triggered by any party, swapping accumulated fees for CLAF on Jupiter and

burning the acquired CLAF. Treasury's `mint_emission` mints the epoch's CLAF emission to the distribution PDA. Participants may then `claim_reward` from the distribution PDA at their convenience.

The data flow is non-trivially asynchronous: settlements may happen at any time within an epoch, the buyback may be triggered at any time after an epoch boundary, and reward claims may be deferred indefinitely. The protocol's correctness does not depend on the timing of these operations, only on their eventual occurrence.

⁂

6. Token Metering and Usage Proofs

The metering of inference usage between two mutually distrusting parties is, in our reading, the central technical problem of permissionless inference markets. We describe the dual-signed proof scheme that we propose for this problem.

6.1 The mutual distrust problem

A user has incentive to under-report token counts; the protocol charges them in proportion to tokens consumed. A provider has incentive to over-report; the protocol pays them in proportion to tokens served. A naive scheme in which either party self-reports is therefore vulnerable in one direction or the other. A scheme in which a third party (an aggregator) reports is also unsuitable; it reintroduces the trusted intermediary whose absence is the protocol's central claim.

Our scheme requires both parties to sign each session's usage proof. The proof contains the user's request hash, the provider's response hash, the token counts (input and output), the price agreed at session opening, and a timestamp. The session is settleable only if both signatures are present. A user who refuses to sign valid proofs forfeits the underlying service (their session times out, their escrow is partially refunded, and their reputation as a counterparty degrades); a provider who refuses to sign forfeits payment.

The scheme does not prevent either party from refusing to participate at all. It does prevent either party from unilaterally distorting the settlement amount. We argue in §11.1 that this property suffices for the protocol's economic security.

6.2 Proof structure

A usage proof is an ed25519-signed message with the following fields.

```
UsageProof {
    session_id:       PubKey,           // Solana account address of session
    user:             PubKey,           // User wallet
    provider:         PubKey,           // Provider wallet
    model_id:         [u8; 32],         // Hash of model identifier
    request_hash:     [u8; 32],         // SHA-256 of canonicalized request
    response_hash:    [u8; 32],         // SHA-256 of canonicalized response
    input_tokens:     u64,              // Tokens consumed on input
    output_tokens:    u64,              // Tokens generated on output
    declared_price:   u64,              // Lamports per million tokens
    timestamp_ms:     u64,              // Unix milliseconds
    user_signature:   [u8; 64],         // ed25519(user_keypair, body)
    provider_signature: [u8; 64],       // ed25519(provider_keypair, body)
}
```

The request and response hashes commit each party to a specific exchange without exposing the content to the chain. The token counts are signed by both parties and therefore agreed by both; disagreement leads to non-settlement. The declared price is the price agreed at session opening and is enforced against the provider's posted offering through Token Metering's `submit_proof` validation.

6.3 Tokenization disagreements

A subtle problem arises if the user and provider disagree on token counts even when both have observed the same request and response. Token counts are a function of the model's tokenizer, which may not be deterministic across implementations. We resolve this with the following rule: at session opening, the provider declares the canonical tokenizer (typically by a hash of the tokenizer's vocabulary file or by a well-known identifier such as `cl100k_base` or `gpt-4o`). The user, by accepting the session, accepts this tokenizer. Both parties compute token counts using this tokenizer and sign their independently-computed result. If the results differ, the lower count prevails for settlement purposes — biasing in the user's favor and discouraging providers from misdeclaring tokenizers.

This rule does not eliminate tokenization games entirely, but it bounds them. A provider who consistently undercounts at the tokenization level loses revenue. A provider who consistently overcounts triggers user-side disagreement and non-settlement.

6.4 The quality factor Q

The reward weight function (§8.3) includes a quality factor Q bounded in the unit interval. We describe how Q is computed.

Q is initialized at 1.0 for newly-registered providers. It decays multiplicatively whenever the provider fails to deliver a response within the session's declared latency budget, when a settlement is challenged and the challenge upheld, when a sampling audit (§11.3) identifies a discrepancy between the provider's declared model and their observed output. The decay rates are constants set at Genesis and are listed in Appendix A. Q recovers multiplicatively with each successful settlement, with a recovery rate calibrated so that providers with sustained good service trend toward Q = 1.0.

Q is intended as a coarse instrument. We do not claim it captures subtle quality differences among providers serving the same model. We claim only that it is sufficient to penalize providers whose service is observably non-functional or fraudulent, and that finer-grained quality discrimination is appropriate to off-chain reputation systems that may emerge above the protocol layer.

⁂

7. Settlement and Capital Flow

The protocol's capital flow is structured around three principles: user funds are non-custodial at all times; settlement is deterministic given a valid usage proof; pressure on CLAF supply through the hardcoded treasury path.

7.1 User deposits and escrow

A user deposits USDC into a Program Derived Address whose seed is ["escrow", user_wallet]. The PDA's authority is the Settlement Escrow program; the program itself controls the release of escrowed funds upon valid usage proofs. Deposits are made via the program's deposit instruction, which requires a user signature.

which requires a user signature. Withdrawals are made via `withdraw`, also requiring a user signature, subject to the constraint that no open sessions encumber the requested amount.

The non-custodial property is verifiable on-chain by anyone with reference to Solana's PDA derivation
S algorithm. Custody exists only as PDA-controlled program state, not as an off-chain record.
th

7.2 Session lifecycle

A session is the unit of metered inference between a specific user and a specific provider for a specific (model, price) pair. Sessions are opened via `open_session`, which reserves an estimated maximum settlement amount against the user's escrow; this reservation prevents the user from spending the same escrow balance with multiple providers concurrently. Sessions are settled via `settle_session` upon submission of a valid dual-signed proof, returning any over-reservation to the user's escrow. Sessions that fail to settle within a 24-hour window expire automatically, returning the entire reservation to the user.

The 24-hour timeout bounds the protocol's exposure to provider unresponsiveness. A provider who serves a request and fails to obtain user co-signature within twenty-four hours loses the revenue from that request. A user who attempts to delay settlement indefinitely (e.g., to deny payment to a provider whose service they actually received) loses the option to obtain future service from that provider, as the provider may decline to engage. The mutual incentives are aligned toward prompt settlement.

7.3 The 97/3 split

Upon settlement, the protocol transfers ninety-seven percent of the settlement amount to the provider's wallet and three percent to the Treasury program's collection PDA. The split is a hardcoded constant; no instruction in the protocol's interface permits its modification. We discuss the rationale for this specific ratio in §8.4.

We note that the 97/3 split is, on its face, significantly more favorable to providers than the take rates of centralized aggregators (typically five to fifteen percent), and it is more favorable still in real terms when one accounts for the protocol's lack of additional charges (no transaction fees beyond Solana's network fees, no monthly minimums, no enterprise tier overhead, no account maintenance fees).

7.4 Automated buyback-and-burn

The Treasury program holds accumulated three-percent fees in USDC. At each epoch boundary, the program's `execute_buyback` instruction — which may be called permissionlessly by any party, with the caller paying Solana network fees — performs the following sequence.

If the treasury balance is below 100 USDC, the instruction returns without action. This threshold prevents inefficient micro-swaps. Otherwise, the program invokes Jupiter aggregator through a cross-program invocation, requesting a swap of the accumulated USDC for CLAF on the open market. The swap is constrained to a slippage tolerance of one percent and a per-transaction volume cap of half a percent of the relevant Jupiter-reported liquidity pool depth; if these constraints would be violated, the swap is broken into multiple sub-operations across the epoch. The acquired CLAF is transferred to the burn address `1nc1nerator11111111111111111111111111111` (a Solana-conventional null address from which no transfer is possible).

The mechanism is fully deterministic given the epoch's settled volume. The execute_buyback instruction is callable by any account; the caller pays gas and receives no privileged outcome. The proceeds route to the hardcoded burn address. Slippage and volume caps are constants in the program's code, immutable after Genesis.

The economic effect of the buyback-and-burn is that CLAF circulating supply contracts in monotonic proportion to protocol usage. We discuss the monetary implications in §8.4.

7.5 Provider bond mechanics

Provider bonds (100 USDC at registration) are held in the Provider Registry's bond PDA. The bond is encumbered as long as the provider is active and may be withdrawn after a seven-day cooldown during which no open challenges exist. Bonds may be slashed by the Settlement Escrow program when a challenge is resolved against the provider; slashing transfers a fixed CLAF amount (30 CLAF, of which 21 to the challenger and 9 to the burn address) from the protocol's distribution PDA to the relevant recipients. Note that the slashing is denominated in CLAF, not USDC; we discuss the reasoning in §11.3.

The bond is the protocol's principal Sybil-resistance mechanism. We discuss its sufficiency under Sybil attack in §11.2.

⁂

8. Monetary Policy

This chapter describes the monetary mechanism of the protocol. Version 1.0 of this paper treated this material in §7, under the heading "Cryptoeconomic Design." We have separated it and expanded it because, on reflection, the monetary mechanism is not merely one design dimension among others — it is the principal mechanism by which the protocol's other properties become economically operative.

8.1 The CLAF emission schedule

CLAF total supply is fixed at one billion tokens. The protocol emits CLAF over a ten-year horizon, with the emission rate halving every two years. The schedule is the following.

Period	Years	Emission	Cumulative
1	0–2	500,000,000 CLAF	500,000,000
2	2–4	250,000,000 CLAF	750,000,000
3	4–6	125,000,000 CLAF	875,000,000
4	6–8	62,500,000 CLAF	937,500,000
5	8–10	31,250,000 CLAF	968,750,000
tail	10+	0	968,750,000

Cumulative emission across all five periods is approximately 968.75 million CLAF. The residual 31.25 million CLAF is permanently uncirculated; the protocol has no mechanism to issue it. This is a deliberate construction: it ensures that the maximum-ever circulating supply is bounded below the nominal total, and the schedule contains no tail-emission instruction after the fifth period.

Within each two-year period, the emission rate is constant per epoch. Epochs are one hour in length; the emission per epoch in Period 1 is therefore approximately 28,538 CLAF per hour, declining by half at the start of Period 2, and so on.

Within each epoch, emission is divided between Provider Pool and Developer Pool in a 70/30 ratio: 70 percent of the epoch emission accrues to verified providers; 30 percent accrues to verified applications and developers.

The schedule is fixed at Genesis. The four parameters that define it (total supply, horizon, halving period, Provider Pool proportion) are constants in the protocol's source code, deployed once and rendered immutable through renunciation of upgrade authority. Emission proceeds according to those constants for the duration of the schedule.

8.2 Halving as monetary lineage

The two-year halving period is a deliberate inheritance from Bitcoin's four-year halving. The choice of two years rather than four reflects the shorter horizon over which AI inference markets are reorganizing, but the underlying logic is identical: a publicly known, mechanistically enforced schedule of supply contraction creates a credible commitment to fixed supply that cannot be coordinated away by interested parties.

We note that the credible commitment is the entire point. The absolute value of the supply cap matters less than its publicness and its inviolability. A protocol that promised one billion CLAF and later issued a second billion under "governance vote" would have no fixed supply at all, regardless of any claim to the contrary. A protocol that promises one billion CLAF and renders that commitment mechanically unforgeable has fixed supply in the only sense that matters: no one can change it.

We also note that the halving mechanism creates a specific economic structure across time. Each halving discontinuously reduces the rate at which new supply enters circulation. If protocol usage is non-decreasing across a halving boundary, the effective real-terms inflation rate falls discontinuously at each boundary. Combined with the buyback-and-burn (§8.4), this produces a token whose supply trajectory is approximately stairstep-deflationary in expectation — a feature Bitcoin-literate participants will recognize, and a feature that we argue in §9 is essential to the cost-subsidy game's sustainability.

8.3 The price-relative subsidy mechanism

The Provider Pool reward weight for provider i in any epoch is

$$W_i = T_i \times (P_{avg}/P_i) \times Q_i$$

where T_i is the provider's verified token volume in the epoch, P_{avg} is the network-wide volume-weighted average price in the epoch, P_i is the provider's own volume-weighted price in the epoch, and Q_i is the quality factor bounded in [0, 1] described in §6.4. The provider's share of the epoch's Provider Pool emission $W_i / \Sigma_j W_j$.

Version 1.0 of this paper presented the P_{avg}/P_i term as a fairness adjustment intended to reward providers who pass consumer surplus to users. This presentation was an understatement of the term's function. We state its function plainly here.

The P_{avg}/P_i term is the protocol's monetary subsidy of supply. It enables providers to rationally operate below their immediate USDC marginal cost — that is, to lose USDC on every transaction — while accumulating CLAF rewards whose expected value, conditional on protocol success, exceeds the USDC loss. This is the structural equivalent of Bitcoin's block reward subsidy of miners. Bitcoin miners do not, in expectation, profit from individual blocks at the marginal cost of electricity; they accept short-term operating losses because they expect the bitcoin they receive to appreciate. ClawFarm providers, in expectation, do not need to profit from individual inference transactions at the marginal cost of GPU amortization; they accept short-term losses because they expect the CLAF they accumulate to appreciate.

The mechanism is asymmetric in its direction. A provider charging $P_i = P_{avg}$ receives a reward weight equal to $T_i Q_i$ — unmodified by the price term. A provider charging P_i below P_{avg} receives a weight strictly greater than $T_i Q_i$. A provider charging P_i above P_{avg} receives a weight strictly less. The mathematics incentivize price competition specifically; they do not incentivize volume competition independent of price.

The consequence at network equilibrium is that the protocol's prevailing price for inference is driven, by the reward structure itself, below the price that would obtain in an unsubsidized market clearing at marginal cost. CLAF emission funds the gap. The gap, in turn, makes the protocol's inference cheaper than any centralized alternative can match: a centralized provider cannot price below its own marginal cost, because it has no mechanism to subsidize itself; the protocol can, because emission is its subsidy mechanism.

We treat the formal analysis of the resulting equilibrium in §9. We note here only that the construction is reflexive — provider behavior depends on expected CLAF appreciation, which depends on protocol usage, which depends in part on the prices providers charge — and that reflexive constructions can succeed (Bitcoin) or fail (many other attempts). We address the failure mode candidly in §9.4.

8.4 Automatic buyback-and-burn

The protocol charges a three percent fee on every settlement. The mechanism by which this fee accumulates and is converted into a contraction of CLAF supply is described in §7.4; we discuss here its monetary significance.

The buyback-and-burn removes CLAF from circulation at a rate strictly proportional to protocol usage. Combined with the fixed emission schedule, this creates a token whose circulating supply is, in expectation and over horizons consistent with the protocol's emission window, monotonically decreasing in the protocol's economic activity. We describe the mathematics.

Let U_t denote the protocol's settled USDC volume in epoch t, and let $P_{\text{CLAF},t}$ denote the CLAF price in USDC at epoch t. The CLAF burned in epoch t is approximately $B_t = 0.03\,U_t / P_{\text{CLAF},t}$, subject to liquidity constraints. The CLAF emitted in epoch t is a fixed E_t determined by the schedule. Net supply change is $E_t - B_t$.

Net supply contracts ($B_t > E_t$) whenever $U_t > E_t P_{\text{CLAF},t} / 0.03$. At Period 1's emission rate (28,538 CLAF/hour) and at hypothetical CLAF prices of \$0.001, \$0.01, \$0.10, and \$1.00, the breakeven USDC volumes per hour are respectively approximately \$951, \$9,513, \$95,127, and \$951,267. The protocol enters a net-deflationary regime — sustained burn exceeding emission — once hourly settled volume crosses these thresholds at the corresponding price levels.

This property is the second half of the cost-subsidy game's sustainability argument. CLAF emission subsidizes provider losses during bootstrap; buyback-and-burn ensures that the CLAF those providers accumulate is appreciating in real terms as protocol usage grows. The two mechanisms together close the loop.

8.5 Why zero premine is load-bearing

The protocol allocates the full 1B CLAF to two emission pools at Genesis. Distribution follows the fixed schedule: 70% to Provider Pool and 30% to Developer Pool over the ten-year emission horizon.

The schedule releases supply only through verified protocol activity; circulating CLAF enters the market through Provider Pool and Developer Pool claims.

This commitment is required for the cost-subsidy game. Providers accept short-term USDC losses in expectation of CLAF appreciation. For that expectation to be rational, the market for CLAF must clear at prices that reflect protocol usage rather than allocation-cliff selling pressure.

The emission schedule replaces the typical premine model. New supply is released over time to providers and developers earning protocol activity. The only CLAF in circulation at any moment is CLAF earned through verified protocol activity by participants who have demonstrated commitment by earning it. In this construction, CLAF price expresses the market's assessment of expected future usage.

This property is achieved through the program's allocation and emission mechanics. A conventional allocation model creates overhang in expectation, because the market anticipates subsequent distributions, grants, or discretionary changes. ClawFarm's supply path is instead expressed through two emission pools and a fixed release schedule, so participants can evaluate the monetary mechanism from deployed rules rather than institutional promises. This is the first instance in this paper of a property whose necessity follows from the protocol's design rather than from its authors' preferences. There will be others, treated in §12.

⁂

9. The Cost-Subsidy Game

We give a formal treatment of the strategic interaction among providers that the protocol induces. The treatment is sketched rather than complete: we establish the structure of the game, the existence of pricing equilibria below marginal cost, the conditions under which such equilibria are sustainable, and the failure modes. Detailed equilibrium proofs under specific functional forms are deferred to a companion technical note.

9.1 Setup

Consider a single epoch with N providers, each indexed by i. Provider i has marginal USDC cost c_i per million tokens served (reflecting GPU amortization, electricity, and operational overhead for Layer C; access price for Layer A; subscription value foregone for Layer B). Each provider selects a price P_i at which to offer service. Token volume T_i served by provider i in equilibrium is a function of all providers' prices, the routing engine's response to those prices, and exogenous demand D. The provider's USDC payoff is

$$\Pi^{USDC}_i = 0.97\, T_i (P_i - c_i)$$

and the provider's CLAF payoff (in CLAF units) is

$$\Pi^{CLAF}_i = (W_i / \Sigma_j W_j) \times 0.7\, E_{epoch}$$

where $W_i = T_i (P_{avg} / P_i) Q_i$ as in §8.3, and E_{epoch} is the epoch's total CLAF emission. The provider's total payoff is the USDC payoff plus the expected USDC-equivalent value of the CLAF payoff:

$$\Pi_i = \Pi^{USDC}_i + E[P_{CLAF,future}] \cdot \Pi^{CLAF}_i$$

The discount factor on the CLAF payoff reflects both the time value of money and the provider's subjective probability that the protocol succeeds.

9.2 Pricing below marginal cost

A provider may rationally choose $P_i < c_i$ — operating at an immediate USDC loss — if the CLAF payoff at that lower price compensates. The first-order condition for the provider's optimal price equates the marginal USDC loss from a price reduction against the marginal CLAF gain. Solving (with caveats deferred to the technical note), the optimal price P_i^* satisfies

$$P_i^* = c_i - k \cdot E[P_{CLAF}] \cdot f(P_{avg}, Q_i, N)$$

where k is a constant absorbing the emission rate and split parameters, and f is an increasing function of P_{avg}, Q_{i}, and (inversely) the effective number of competing providers N. The interpretation is that providers price below cost by an amount proportional to the expected CLAF appreciation: higher expected appreciation, deeper below-cost pricing.

The crucial property of this equilibrium is that it is self-stabilizing in expansion. As more providers enter, N grows, the CLAF reward per provider shrinks, and the optimal below-cost discount decreases. As protocol usage grows, P_{CLAF} rises (via buyback-and-burn), and the optimal below-cost discount increases. The two effects produce an entry-and-pricing dynamic in which provider density adjusts to the protocol's economic state.

It is also self-stabilizing in contraction, with a different sign. If protocol usage falls, P_{CLAF} falls, the below-cost subsidy contracts, and providers exit. This is the failure mode of §9.4.

9.3 Early-miner asymmetry

The earliest providers experience an asymmetry that is, in our reading, the central economic feature of the protocol's bootstrap phase. We describe it.

At the protocol's launch, the provider population is small. For each provider, the term $\Sigma_{\text{j}} W_{\text{j}}$ in the denominator of their reward share is correspondingly small, and each provider's share of the epoch's emission is correspondingly large. If protocol usage at launch is also low (which we expect), the absolute USDC value of the reward share is small as well — but the share itself, measured in CLAF units, is large.

A provider who serves volume in the first months of the protocol's operation accumulates CLAF at a per-token rate orders of magnitude above what later providers will receive. If the protocol's usage subsequently scales and CLAF appreciates, this early-stage CLAF accumulation produces a wealth distribution among providers that is heavily skewed toward the earliest entrants. This is, again, the Bitcoin precedent: the earliest miners accumulated bitcoin at trivial cost, and those who held that accumulation through the subsequent appreciation became disproportionately wealthy in proportion to their early participation rather than their absolute contribution.

We are explicit about this property because it is, on the protocol's design, intended. The early-miner asymmetry is the mechanism by which the protocol attracts participation when its present economic value is small. It is not a flaw to be designed out; it is the protocol's primary recruiting mechanism for the bootstrap phase.

9.4 Failure modes

The cost-subsidy game can fail in three ways, which we describe candidly.

Cold-start failure. If the protocol fails to attract sufficient initial provider participation in the first six to twelve months — say, fewer than one hundred providers across all three layers — the network's volume-weighted statistics become degenerate, the routing engine's price discovery becomes thin, and individual users observe a market that fails to provide reliable service. Subsequent participants observe this outcome and decline to join. The protocol enters a contractionary spiral and does not recover. We assess the probability of this failure mode as material — not negligible. We have designed the early-miner asymmetry to mitigate it but cannot eliminate it.

CLAF price collapse. If CLAF fails to appreciate as protocol usage scales — for instance, because the market discounts the protocol's commitments more heavily than the design assumes, because external macroeconomic conditions depress all crypto-asset prices simultaneously, or because some unforeseen vulnerability is discovered and exploited — the below-cost provider pricing equilibrium of §9.2 unravels. Providers who priced below marginal cost on the expectation of CLAF appreciation will, after sustained price weakness, exit. The protocol's prices rise, demand falls, and the protocol enters the contractionary spiral.

Centralized provider concentration. If a single provider (or a small cartel of providers) achieves dominant volume share and sustains it, the protocol's market discipline degrades. The provider can charge above the competitive equilibrium price; the price-relative subsidy term ceases to bind effectively (because the dominant provider sets P_{avg}); and the protocol's economic value to users diminishes. We have designed the bond requirement and the reward function to discourage this outcome, but at very high market shares the design may not suffice.

We do not claim that any of these failure modes is improbable. We claim only that, conditional on the protocol's general thesis being correct, the design described in this paper is the construction with the highest expected probability of success that we have been able to identify. The probability of success is, candidly, well below one.

⁂

10. Routing Engine

The protocol does not specify a single routing engine, nor does it require one. We describe the architectural envelope within which any routing engine must operate, the three routing modes the reference implementation offers, and the protocol's relationship to off-chain routing services.

10.1 Routing as an off-chain function

The protocol's on-chain programs are concerned with the verification and settlement of inference, not with the selection of which provider serves which request. Selection is an off-chain function. A user wishing to obtain inference must, at some point, decide which provider to engage; the protocol does not prescribe how this decision is made.

This separation is deliberate. On-chain routing would impose Solana-level transaction costs on every match decision, would force routing logic to be globally consistent across all participants, and would create a single point of capture for routing policy. Off-chain routing avoids each of these problems: routing services compete on quality, users may operate their own routing or use third-party services, and the protocol's settlement guarantees do not depend on which routing service was used.

10.2 Reference routing modes

The reference implementation, which we release as open source under MIT license, offers three routing modes.

Eco mode selects the provider with the lowest declared price for the requested model, subject to a minimum quality factor (Q ≥ 0.85 by default) and a minimum recent volume (the provider must have served at least 1,000 tokens in the prior 24 hours). This is the appropriate mode for cost-sensitive workloads where latency and quality differences within the eligible provider set are acceptable.

Auto mode selects from the eligible provider set weighted by inverse price, inverse latency (measured over the prior 24 hours), and quality factor. The weights are exposed as user-configurable parameters; the defaults are price 50%, latency 30%, quality 20%. This is the appropriate mode for general production workloads.

Premium mode restricts selection to providers with the highest Q factors (Q ≥ 0.98) and the highest recent volumes (top decile), with price as a tiebreaker. This is the appropriate mode for workloads that require predictable service characteristics.

The reference implementation is one possible routing engine. Users may run their own routing logic, use third-party services, or compose custom strategies. The protocol's settlement guarantees are identical regardless of routing choice.

10.3 Routing service competition

Competing routing services create market discipline against routing manipulation. A routing service that systematically biases against certain providers will be observable in its output: users comparing routing services will detect the bias and switch. A routing service that captures fees in addition to the protocol's three percent will be observable in its pricing: users comparing the cost of routing services against direct routing will switch. The protocol does not legislate routing service behavior; it makes such behavior verifiable and switchable.

We acknowledge that the protocol's reliance on off-chain routing imports some of the same coordination problems as centralized aggregators. A routing service that captures the majority of user flow exerts substantial influence on provider economics. We treat this as a partial limitation: the protocol's settlement layer is permissionless and credible, but the routing layer above it may, in practice, be more concentrated than the protocol designers would prefer. We discuss possible mitigations — including on-chain routing experiments and reputation-based routing — in §15.

⁂

11. Security Analysis

We analyze the protocol's behavior under the threat models specified in §3.3, with attention to economic security in addition to traditional cryptographic security.

11.1 Settlement integrity under metering disagreement

The dual-signed proof scheme of §6 ensures that no settlement can occur without the cooperation of both parties. We analyze the possible deviations.

A provider who unilaterally over-claims tokens served cannot obtain settlement: the user, observing the discrepancy, will not co-sign. The provider has lost the resource cost of serving the request but suffers no further penalty unless a challenge is brought. A provider who unilaterally under-delivers (serves a cheaper model than declared, or truncates responses) likewise faces user non-cooperation, with the additional risk of a sampling-audit-triggered challenge.

A user who unilaterally refuses to co-sign a valid proof denies the provider settlement. The provider has served the request and is unpaid. The user has obtained free inference for a single session. However, the provider's response need not be returned in cleartext until the user has co-signed an outline of the request; the protocol-recommended exchange protocol (described in the reference implementation documentation) provides that the user sends a request commitment, the provider returns an encrypted response with a hash and a session-settlement-pending status, and the user releases the decryption key only upon co-signing. Under this discipline, free inference is technically possible but operationally costly to users: their session reputation degrades, future providers may decline to serve them, and the protocol-level interest in maintaining good standing exceeds the value of single-session free inference for any user who intends repeat use.

11.2 Sybil resistance under bond

An adversary may register multiple provider identities, posting 100 USDC per identity, to capture disproportionate emission share. We analyze the conditions under which this is profitable.

For a Sybil attack to be profitable, the marginal CLAF emission captured by an additional identity must exceed the marginal cost of acquiring it (100 USDC bond plus opportunity cost of operating the identity). The reward weight function's structure makes this difficult: an additional identity must serve real volume to accumulate T_i, and the reward share is a function of price-weighted volume relative

to the network. Two adversarial identities operated by the same actor cannot serve the same physical capacity twice — they share the underlying GPU or API access. Sybil registration without underlying additional capacity therefore captures no additional emission; only Sybil registration with genuinely independent capacity does, and at that point, the Sybil is indistinguishable from honest provider entry.

The remaining attack vector is reputation laundering: an adversary may register multiple identities to evade quality-factor decay on any single identity. The protocol does not eliminate this; it bounds it through the 100 USDC bond cost per identity, the seven-day bond withdrawal cooldown, and the multiplicative quality decay (which makes laundering only valuable across long time horizons, by which point sampling audits have likely fired). We assess this as adequate but not optimal protection; future versions of any protocol in this design space might consider stronger reputation mechanisms.

11.3 Sampling audits and challenger-driven enforcement

The protocol supports challenges as described in §5.2 and §7.5. We describe the auditing layer that we propose operate above the protocol.

The reference implementation includes a sampling auditor: a process that monitors settlements, selects a small random sample (we propose 1%), reissues the underlying request to a known-good reference provider for the declared model, and compares results. If the result diverges materially from what the audited provider declared (different output, missing portions, lower quality), the auditor posts a challenge against the settlement.

Challenges are economically self-funding through challenger reward: a successful challenge slashes 30 CLAF from the challenged provider, of which 21 are paid to the challenger and 9 are burned. The 2 USDC challenge bond is refunded on success and forfeited on failure. The economic balance is such that auditing is profitable in expectation if and only if real fraud exists at a rate above some threshold; honest providers face challenges only stochastically, with refunded bonds and no slashing.

The protocol does not designate a single auditor. Any party may operate one. The reference implementation's auditor is one of many possible; we expect the auditor population to mirror, in structure, the provider population — many small operators rather than a small set of dominant ones.

11.4 Collusion at small scale

In a small provider population, providers may attempt to collude on prices, raising prices uniformly to extract surplus from users. We analyze why this is destabilizing in the protocol's specific design.

Suppose all N providers collude to set $P_i = P^{cartel} > c$. The volume-weighted average price P_{avg} equals P^{cartel}, and the reward weight term (P_{avg} / P_i) equals 1 for all providers. The cartel jointly extracts maximum USDC surplus from users.

A defection from the cartel — a single provider charging $P^{defect} < P^{cartel}$ — earns the defector both a larger volume share (since they undercut the market) and a larger CLAF reward weight (since their $P_i < P_{avg}$). The cartel-incentive structure is therefore strictly dominated: defection yields strictly higher payoff than cooperation, at every level of CLAF expected appreciation. Standard cartel coordination problems apply with additional force.

We do not claim the protocol is immune to all forms of collusion — extralegal mechanisms, off-chain agreements, side payments — can in principle sustain collusion despite the protocol's incentive structure. We claim only that the on-chain incentive structure works against collusion rather than for it, and that the protocol's transparency (all settlements are on-chain, all prices observable) makes coordinated deviation detectable.

11.5 Underlying chain dependence

The protocol's security properties are upper-bounded by Solana's security. If Solana suffers consensus failure, the protocol fails with it. We treat this as an external risk and do not design around it.

We note that the protocol's economic mechanisms — the cost-subsidy game, the buyback-and-burn — depend on Solana-native infrastructure (Jupiter aggregator for swaps, SPL token standard for CLAF, ed25519 signatures for usage proofs). A successor protocol on a different chain would require reimplementation of each of these dependencies. The protocol is not portable. This is a feature insofar as it forecloses the temptation to "migrate" in response to issues that should instead force the protocol's failure; it is a limitation insofar as Solana's properties may not, over the protocol's full emission horizon, remain favorable.

11.6 Model-identity verification

The protocol cannot verify that a provider's served output came from the model the provider declared. A provider may declare service of DeepSeek V4 Pro and actually serve DeepSeek V4 Flash, or any other cheaper model. The quality factor Q and sampling audit provide partial mitigation — egregious model substitution will fail audits — but subtle substitution (a slightly smaller variant of the declared model) may evade detection.

We do not claim this is solved. We note that the problem is shared by all inference markets, including centralized aggregators; the difference is that aggregators may have contractual recourse against providers, while the protocol does not. We view this as a candidate direction for future cryptographic work (verifiable inference, trusted execution environments) and discuss it in §15.

⁂

12. Structural Necessities

This chapter argues that several of the protocol's design choices — pseudonymous authorship, zero allocation, immutability, absence of governance — are not preferences but necessities. The argument is unconventional. We make it carefully.

12.1 Non-custodial settlement

The protocol's user funds are held in PDAs whose authority belongs to a Solana program rather than to any private key. We treated the mechanism in §7.1. We argue here why the property is necessary.

A protocol whose user funds are custodial in any form — held by a company, multisig, foundation, or other party — depends on that party's continued cooperation for fund security. The party may be subpoenaed, may be insolvent, may be coerced, or may simply choose to misappropriate. None of these failures is hypothetical; the history of crypto-asset custody includes examples of each. A custodial protocol cannot make the credible commitments to user fund security that the protocol's other functions require.

Non-custodial settlement makes those commitments mathematical. Funds are held in PDAs derivable by anyone, controllable by no one. The protocol's authors cannot misappropriate them because no path exists through the protocol's interface to do so. The cost of this property is some loss of flexibility — a custodial protocol could, for instance, reverse demonstrably fraudulent settlements, while the present protocol cannot. We accept this cost.

12.2 Renounced upgrade authority

The protocol's programs are deployed with renounced upgrade authority. After Genesis, program code is fixed by the deployed account state. We treated the mechanism in §13. We argue here why the property is necessary.

A protocol with retained upgrade authority depends on that authority's continued exercise in good faith. The authority-holder may be coerced, may change their mind, or may be subject to legal process that compels them to modify the protocol against the participants' interests. A protocol that promises certain rules but retains the ability to change them has not, in the contractual sense in which participants must rely on the rules, promised anything.

Renouncing upgrade authority makes the protocol's rules mathematical. The four parameters that define the monetary policy (total supply, emission horizon, halving period, Provider Pool proportion) are constants in the deployed bytecode. The 97/3 split, the 3% buyback-and-burn target, the 100 USDC bond, and the slashing amounts are constants. Subsequent transactions execute against those constants. The cost of this property is the foreclosure of future improvements; if a flaw is discovered after Genesis, the protocol cannot be patched. We accept this cost.

12.3 Absence of governance

Protocol parameters are fixed at Genesis and apply for the duration of the emission schedule. The monetary constants, settlement split, bond amount, slashing amounts, and treasury path are encoded as program behavior rather than as a process for later modification. We argue here why the property is necessary.

Governance, even when nominally decentralized, concentrates over time. The history of token-governed protocols includes substantial examples of voter participation collapsing to single-digit percentages, of governance tokens accumulating in the hands of a small set of large holders, of "decentralized" protocols being effectively controlled by a handful of parties. A protocol whose rules can be

changed by governance has rules that, in practice, the controlling parties can change. The protocol cannot make credible commitments beyond what those parties will tolerate.

Parameter fixation makes the protocol's commitments unconditional. The emission schedule, treasury path, and settlement split are not inputs to a recurring decision process; they are program constants. The protocol does what it does, and that is all. The cost is the loss of any mechanism to respond to circumstances unforeseen at Genesis. We accept this cost.

T
tr
c
is
W

12.4 Structural neutrality across supply sources

The protocol does not record, and is technically incapable of recording, which of the three supply layers any given provider draws from. We discussed the mechanism in §4.4. We argue here why the property is necessary.

A protocol that recorded provider supply source would, by virtue of holding that record, become a target for legal process from upstream parties whose terms of service some providers may violate. The protocol's operator could be subpoenaed to disclose Layer B participants. The disclosure would be possible because the information would exist. The legal target would be locatable because the operator would have a location. Layer B participation would be conditional on the operator's willingness to absorb the resulting risk, which would limit Layer B scale, which would limit the protocol's economic value during the period in which Layer B's arbitrage window is open.

Structural inability to record makes the protocol's neutrality unconditional.
Structural inability to record makes the protocol's neutrality durable. Supply-source information is absent from protocol records, so providers in any layer participate on identical terms.

terms; the protocol's economic mechanism applies identically to all. We accept the cost — which is that some participants may use the protocol for activities that upstream parties would object to — because the alternative is a protocol whose scale is bounded by the most legally-conservative supply source.

12.5 Zero allocation

We argued the necessity of zero allocation in §8.5. We restate the argument here for completeness.

The cost-subsidy game (§9) requires provider behavior that is rational only if CLAF appreciation is credibly expected. Credible expectation requires that the market clear at prices reflecting protocol usage, unadulterated by overhang from non-usage holders. Zero allocation removes the overhang. Any allocation — to team, investors, advisors, foundation, anyone — introduces overhang and undermines the credibility on which the cost-subsidy game depends.

Zero allocation is, in the protocol's economic design, load-bearing. It is not a generosity or austerity. It is the construction's structural foundation.

12.6 The 3% fee buyback-and-burn

The protocol's three percent fee is automatically converted into CLAF buyback-and-burn. We argued the mechanism in §7.4 and the monetary effect in §8.4. We argue here why the property is necessary.

The protocol's monetary credibility requires that participants believe the protocol's stated mechanisms will execute as designed. The buyback-and-burn is the principal mechanism by which CLAF appreciates with usage; if it could be redirected — to a foundation, to a development fund, to a marketing budget — participants would correctly discount its effect, and the cost-subsidy game would be correspondingly weaker.

Automation under cross-program invocation, with hardcoded constants and burn destination, makes the property unconditional. The fee will be buyback-burned as long as the protocol operates. The cost is that the fee cannot be redirected to funded development capacity; the benefit is a fee mechanism whose credibility is bounded by program execution rather than discretionary stewardship.

b
b
h
d
cr

12.7 Pseudonymous authorship as structural necessity

The authors of this paper write under the pseudonym C. Wren. The pseudonym is not a marketing device. It is a structural feature of the protocol's design. We explain its necessity here.

A protocol that performs the functions described in this paper — admitting provider capacity from multiple sources without verification, committing to a fixed monetary schedule that cannot be modified, operating under a cost-subsidy game whose sustainability requires credible permanence of all stated parameters — places severe constraints on the organizational form that can implement it. We enumerate the constraints.

First, the protocol cannot have an identifiable issuer for purposes of securities law, because such an issuer would face disclosure obligations that would require it to characterize the protocol's operation in terms that may not survive regulatory scrutiny. The protocol's structural neutrality across supply sources, in particular, is incompatible with the disclosure profile expected of a regulated issuer. A protocol whose authors are not identifiable as issuers — because they are not identifiable as authors of a security in the regulatory sense — does not face this obligation.

Second, the protocol cannot have an identifiable promoter for purposes of the same body of law. A known person making forward-looking statements about a token's expected appreciation, operating channels through which buy-side demand is encouraged, or holding significant token quantities in their own name, falls within the regulatory definition of a promoter. The protocol's cost-subsidy game depends on participants forming expectations about CLAF's future value, but those expectations must form without identifiable promotion, or the protocol's authors become the legal target of any subsequent market action. Pseudonymous authorship resolves this by separating the authorship function from the promotion function entirely — the authors do not promote, because they cannot be reached to be asked to.

Third, the protocol cannot have an entity capable of being captured by external pressure. A registered company, even one with renounced control over protocol parameters, retains employees, bank accounts, office addresses, and legal counsel — all of which are pressure surfaces. A subpoena to the company can demand disclosure of provider identities, transaction details, or operational practices, even where the protocol itself does not record them, because the company's employees may possess inferable information. Pseudonymous authorship without an associated entity eliminates this surface entirely.

Fourth, and most consequentially, the protocol cannot have authors who themselves stand to benefit from the protocol's economic activity in ways legible to the market. The cost-subsidy game requires participants to believe that CLAF's price reflects the protocol's future usage and not the actions of its authors. Authors known to hold substantial CLAF, known to operate businesses that benefit from the protocol's success, or known to coordinate with parties that do, undermine this belief. The market correctly discounts CLAF in such cases. The discount, propagated back through the cost-subsidy game, reduces provider participation, which reduces protocol usage, which validates the discount. The equilibrium con-

tracts. Pseudonymous authorship without identifiable economic benefit to the authors is the only form under which the cost-subsidy game's equilibrium can be the expansionary one rather than the contractionary one.

We note that this argument is structurally identical to the argument for Bitcoin's pseudonymous founder. Satoshi Nakamoto's anonymity was not, in our reading, a personal preference or a regulatory evasion. It was the unique configuration under which Bitcoin's monetary commitments could be credible. A known founder holding five percent of Bitcoin's supply would have made Bitcoin a different asset — one whose monetary properties were contingent on the founder's continued cooperation, rather than guaranteed by the protocol's design. The same logic applies, with the same force, to any subsequent protocol attempting a Bitcoin-like monetary construction.

C. Wren is the pseudonym under which the v1.0 and v2.0 papers are published. The protocol's economic exposure is defined by participation, not authorship: rewards are earned by providers and consumers under the same formulas used by all other addresses. Program parameters are fixed by the deployed bytecode, and emission proceeds according to the schedule encoded at Genesis. This is the structural condition under which a Bitcoin-like monetary construction can be attempted for AI inference.

o
li
e
te
k
a
te

ti

⁂

13. Implementation

We describe the protocol's deployment, the Genesis transaction's contents, and the operational implications of immutability.

13.1 Genesis transaction

The Genesis transaction is the single Solana transaction in which the protocol's four programs are deployed and their parameters fixed. The transaction's atomicity is consequential: either all four programs deploy with their final parameters, or none do, and any failure aborts the entire deployment. We describe the transaction's contents.

First, the four programs (Provider Registry, Settlement Escrow, Token Metering, Treasury) are deployed via standard BPF program loading. The bytecode of each program is published in advance and verifiable; the deployment transaction's program data must match the published bytecode hashes.

Second, the protocol's Mint Authority for CLAF is set to the Treasury program's PDA; mint authority is held by the program-derived address (PDA) and invoked only through the Treasury program's mint instruction, gated to execute along the emission schedule. The total CLAF ever mintable cannot exceed the schedule's cumulative emission.

Third, the four programs' upgrade authorities are renounced. The upgrade authority field of each program account is set to the burn address. After this point, program code is fixed by the deployed account state.

Fourth, the deployer's wallet, the Solana wallet from which the Genesis transaction is signed, has its remaining balance (after accounting for transaction fees and program-account rent-exempt deposits) transferred to the burn address. The deployer wallet is expected to remain inactive after Genesis; that inactivity is verifiable from subsequent on-chain history.

Fifth, the Genesis event is logged with a publicly verifiable timestamp and the bytecode hashes of the four programs. After Genesis, the protocol is operational, immutable, and unowned.

13.2 Operational implications of immutability

Immutability has operational implications that participants should understand.

The protocol cannot be paused through an administrative instruction. If a critical bug is discovered after Genesis, operation follows the deployed bytecode until a successor protocol is deployed by another author. Bugs that allow value extraction will be extracted from. Bugs that prevent the protocol from operating will simply prevent operation until a successor protocol exists.

The protocol cannot be upgraded. New features cannot be added. Performance optimizations cannot be applied. Compatibility with new versions of underlying infrastructure (Solana protocol changes, Jupiter aggregator updates) is bounded by what the deployed programs already support; if Jupiter's interface changes incompatibly, the buyback-and-burn fails until and unless Jupiter maintains backward compatibility.

The protocol's parameters are fixed forever. If the 100 USDC bond proves too small in practice, it remains 100 USDC. If the 3% fee proves too small to support buyback-and-burn at desired pressure, it remains 3%. If the two-year halving period proves to be the wrong pace, it remains two years.

We accept these implications because the alternative — a protocol whose parameters can be modified — sacrifices the credible commitments on which the protocol's economic design depends. A flawed protocol with credible commitments is, in our reading, more valuable than an updatable protocol with no credible commitments at all.

13.3 Reference implementation

The reference implementation of all four programs is released under MIT license at `github.com/clawfarm-protocol/programs`. The implementation is in Rust, using the Anchor framework. The release includes test suites, deployment scripts, and a sample client SDK in TypeScript and Python.

We note that the reference implementation is one possible implementation. The protocol's specification is given in this paper and in the on-chain bytecode; any party may reimplement the client side using any tooling. We expect, over time, multiple client implementations to emerge.

⁂

14. Extended Economic Analysis

We extend the analysis of §9 to several specific questions of practical interest.

14.1 Provider economics at protocol launch

We sketch the per-provider economics in the protocol's first six months under specific scenarios. The scenarios are illustrative and depend on assumptions we mark explicitly.

Scenario: Layer C provider, single H100, DeepSeek V4 Flash, modest CLAF expectation. A provider operates one H100 GPU at approximately $2.69/hour rental cost (or equivalent owned-hardware amortization). Marginal cost per million tokens served, assuming sustained throughput of 200 tokens/second, is approximately $3.73/MTok all-in. Suppose the provider prices at $0.30/MTok — substan-

Monthly served volume: approximately 518 million tokens. Monthly USDC revenue: 518 × $0.30 × 0.97 · $151. Monthly USDC marginal cost: approximately $1,963. Monthly USDC payoff: ·$1,812.

Suppose the network has 50 active providers with comparable price and volume. The provider's CLAF reward weight share is approximately 1/50 of the Provider Pool emission for the epoch. Monthly emission to Provider Pool at Period 1 rate: 70% of 28,538 × 24 × 30 · 14.38 million CLAF. The provider's share: approximately 287,600 CLAF per month, before the price-relative term.

If the provider's price is 1/3 of network average (P_avg ≈ $0.90, P_i = $0.30), the price-relative term is 3.0, lifting their weight by 3×. Adjusted share: approximately 862,800 CLAF per month.

For this provider to break even on a USDC basis, the CLAF price must reach $1,812 / 862,800 ≈ $0.0021 per CLAF. If the provider expects CLAF to reach $0.01 within twelve months — a not-implausible level for a protocol with successful bootstrap — their twelve-month total payoff is (12 × −$1,812) + (12 × 862,800 × $0.01) = −$21,744 + $103,536 = $81,792.

The same calculation at CLAF = $0.001 yields total payoff −$21,744 + $10,353 = −$11,391 (loss). At CLAF = $0.0005, total payoff −$21,744 + $5,177 = −$16,567 (larger loss). The provider's expected payoff is positive if and only if their expected average CLAF price exceeds approximately $0.0021 over the relevant horizon.

This sensitivity is the central economic feature of the bootstrap phase. Provider participation depends on provider beliefs about CLAF appreciation. The protocol's design — zero allocation, fixed schedule, automated burn — exists to make those

beliefs credible. If the design succeeds in producing credible beliefs at low CLAF prices, the cost-subsidy game proceeds. If it does not, providers exit, and the protocol fails.

14.2 Aggregate market sizing

We attempt a rough sizing of the protocol's addressable market across the three supply layers.

The total global AI inference market, by spend, is estimated at $35 billion annually as of mid-2026 (Menlo Ventures), with growth rates of 3-5× year-over-year. Open-weight inference accounts for approximately 4% of revenue but 20% of token volume on managed-aggregator platforms (OpenRouter / a16z State of AI study, 2026). The price differential between open and closed inference at comparable quality is approximately 5× for non-frontier tasks.

Within this market, the protocol's three layers face the following sizing.

Layer A (closed-source resale). The aggregator market — OpenRouter, Together AI, Fireworks, and competitors — is approximately $1-3 billion in directly-routed inference annualized as of mid-2026. The protocol's Layer A competes with aggregators on margin (3% vs. 5-15%) and on permissionlessness. We estimate the protocol's plausible Layer A take, at maturity, in the range of $100-500 million annual settlement volume (5-25% of the aggregator market).

Layer B (subscription credit secondary market). The aicredits.co data point — $20 million transacted with 90% repeat rate as of early 2026 — suggests a current secondary market of $50-200 million annual volume. The protocol mechanizes this market on-chain. We estimate Layer B at $50-300 million annual settlement volume, with substantial uncertainty due to the six-to-eighteen-month window before laboratory countermeasures.

Layer C (open-weight self-hosted). The current self-hosted inference market is small in spend but growing rapidly; open-weight production-grade inference deployments are emerging at scale across enterprise CIOs, regulated industries, and cost-sensitive applications. We estimate Layer C at $500 million to $5 billion in 2027-2028, growing toward $15-40 billion by 2030 if open-weight inference quality continues its current trajectory. The protocol's plausible Layer C take, at maturity, in the range of 5-15% of this market.

The total protocol settlement volume at maturity, summing across layers, plausibly ranges from $1 billion to $5 billion annualized in a base case, with substantially higher figures in a bull case where open-weight inference becomes the dominant inference modality. We do not predict which case will obtain. We note that the protocol's revenue (3% of settlement) at $1 billion volume is $30 million annually, all of which becomes buyback-and-burn pressure on a fixed-supply token.

14.3 Comparison to comparable protocols

For reference: Bittensor's market capitalization as of mid-2026 is approximately $2.5-3 billion; Render's is approximately $1.5-2 billion; Akash's is in the hundreds of millions. Each of these protocols generates substantially less than $50 million in annualized network revenue. The market is, accordingly, pricing each at a substantial multiple of revenue — consistent with the broader pattern in which decentralized infrastructure tokens trade on a combination of usage, network growth, and narrative.

We make no prediction about ClawFarm's market capitalization at any point. We note only that the protocol's structural features — fixed supply, automatic burn, zero allocation, immutability — produce a token whose supply trajectory is monotonically deflationary in usage, and that the market's historical treatment of similarly-structured tokens (Bitcoin, in particular) has been to reward the deflationary trajectory with substantial multiples of present revenue. Whether the market extends similar treatment to ClawFarm is a question we leave to the market.

⁂

15. Future Work

We identify several directions in which the protocol's design space could be extended. We make no commitment to pursuing any of them; ClawFarm itself is, by construction, fixed at Genesis. The directions below are candidates for separate protocols by separate authors.

15.1 Verifiable inference

The protocol's central technical limitation, identified in §11.6, is the inability to verify that a provider's output came from the declared model. Cryptographic approaches to verifiable inference — zero-knowledge proofs of model execution, trusted execution environments with attestation, multi-party computation across provider sets — are active research areas. A successor protocol incorporating verifiable inference would substantially strengthen the Q factor's meaning and allow more direct quality-based pricing.

15.2 Cross-chain settlement

ClawFarm is Solana-bound. A cross-chain successor — settling inference on multiple chains simultaneously — would broaden the protocol's reach but at substantial complexity cost. We do not pursue this in v2.0 because the security and economic guarantees of cross-chain operations are, in our reading, not yet at parity with single-chain operations.

15.3 Specialized routing protocols

The protocol's reference routing engine (§10) is intentionally simple. Specialized routers — for latency-sensitive workloads, for privacy-preserving workloads, for workloads requiring specific quality guarantees — could be built above the protocol. We expect such routers to emerge organically; no protocol-level support is required.

15.4 Reputation systems

The protocol's Q factor is coarse. Off-chain reputation systems — provider reviews, third-party audit certifications, performance benchmark records — could provide finer-grained quality discrimination. We expect such systems to emerge above the protocol layer, monetized through their own mechanisms.

15.5 Confidential inference

The protocol's request and response hashes commit each party to the exchanged content, but the content itself is not protected from a malicious provider who simply reads it. Confidential inference — through trusted execution environments, homomorphic encryption, or multi-party computation — would address this. The state of the art in 2026 makes confidential inference operationally feasible for specific model classes but expensive in general. We do not address it in v2.0.

⁂

16. Conclusion

We have presented ClawFarm as a protocol for the production and exchange of AI inference, deployed on the Solana blockchain as a set of immutable programs, with monetary policy implemented through automatic emission and buyback-and-burn. We have argued that its design lineage is Bitcoin: that its three-layer supply architecture admits capacity from closed-source resale, subscription credit secondary markets, and open-weight self-hosted inference without distinction; that its cost-subsidy game enables providers to operate below their immediate marginal cost in expectation of CLAF appreciation; and that its load-bearing features — zero allocation, pseudonymous authorship, renounced upgrade authority, absence of governance — are structural necessities rather than ideological commitments, achievable under no alternative organizational form.

We have not claimed that this is the only configuration in which a permissionless inference market could exist. We have claimed only that it is one configuration, that the configuration is internally consistent, and that the conditions under which it can be attempted are present in the AI inference market as of mid-2026 and not before. The window is, in our reading, real but not permanent. The market for open-weight inference is consolidating rapidly; the centralized aggregators are improving rapidly; the regulatory environment for tokenized markets is shifting. A protocol that occupies this design space must do so soon, or not.

We have built the protocol because we believe it should exist. We have deployed it because we believe the conditions for its success are present. We have authored this paper under pseudonym because we believe its success depends on its independence from us. After Genesis, we have no further operational role. The protocol persists, is used, or is forgotten, according to the decisions of its participants. The authors of this paper have no remaining lever to pull.

> *Compute is permissionless. Settlement is automatic. Rewards follow contribution. The rest is left to the participants.*

This is the construction. It exists or it does not. It works or it does not. We have done what authorship can do. The protocol's life, henceforth, is not ours.

⁂

References

[1] Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. bitcoin.org/bitcoin.pdf

[2] Solana Labs. (2024). *Solana Programming Model: Program Derived Addresses*. docs.solana.com

[3] Jupiter. (2025). *Jupiter Aggregator: Cross-Program Invocation Interface*. station.jup.ag/docs

[4] OpenRouter, a16z. (2026). *The State of AI: An Empirical 100-Trillion-Token Study*.

[5] Nagle, F. & Yue, D. (2026). *Open-Source AI Adoption and the Economics of LLM Inference*. MIT Initiative on the Digital Economy.

[6] Bittensor Foundation. (2025). *Yuma Consensus and the Bittensor Subnet Architecture*. bittensor.org/docs

[7] Render Network. (2024). *Burn-and-Mint Equilibrium: Render's Token Economic Model*. rendernetwork.com

[8] DeepSeek. (2026). *DeepSeek V4: Model Card and Technical Report*. huggingface.co/deepseek-ai

[9] Moonshot AI. (2026). *Kimi K2.5: Open-Weight Frontier Reasoning Models*. github.com/moonshotai

[10] Alibaba DAMO Academy. (2026). *Qwen 3.5 Technical Report*. qwenlm.github.io

[11] Hinman, W. (2018). *Digital Asset Transactions: When Howey Met Gary (Plastic)*. Speech, U.S. Securities and Exchange Commission.

[12] Coinbase, Cloudflare. (2025). *x402: HTTP-Native Micropayments for the Agentic Web*. x402.org

[13] Sacra Research. (2026). *OpenRouter Annualized Inference Spend, May 2026*. sacra.com/c/openrouter

[14] AI Credits. (2026). *Marketplace Statistics: Twenty Million in Traded Credits Across 200+ Customers*. aicredits.co

⁂

Appendix A. Genesis Parameters

This appendix lists all constants embedded in the protocol's bytecode. After Genesis, these values cannot be modified.

```
// MONETARY POLICY
CLAF_TOTAL_SUPPLY            = 1_000_000_000     // CLAF (whole tokens)
EMISSION_HORIZON_YEARS       = 10
HALVING_PERIOD_YEARS         = 2
NUM_HALVING_PERIODS          = 5
EMISSION_PERIOD_1            = 500_000_000       // CLAF
EMISSION_PERIOD_2            = 250_000_000
EMISSION_PERIOD_3            = 125_000_000
EMISSION_PERIOD_4            = 62_500_000
EMISSION_PERIOD_5            = 31_250_000
TAIL_UNEMITTED               = 31_250_000        // permanently uncirculated
EPOCH_LENGTH_HOURS           = 1
SUPPLY_SIDE_SHARE            = 0.70
DEMAND_SIDE_SHARE            = 0.30

// SETTLEMENT
PROTOCOL_FEE_BPS             = 300               // 3.00%
PROVIDER_SHARE_BPS           = 9700              // 97.00%
SESSION_TIMEOUT_HOURS        = 24
SETTLEMENT_FINALITY_BLOCKS   = 32                // Solana confirmations

// PROVIDER REGISTRY
PROVIDER_BOND_USDC           = 100               // (in USDC, 6 decimals)
BOND_COOLDOWN_DAYS           = 7
CHALLENGE_WINDOW_HOURS       = 24
CHALLENGE_BOND_USDC          = 2
SLASH_AMOUNT_CLAF            = 30
SLASH_TO_CHALLENGER_CLAF     = 21
SLASH_TO_BURN_CLAF           = 9

// QUALITY FACTOR
Q_INITIAL                    = 1.0
Q_DECAY_LATENCY              = 0.98              // per missed deadline
Q_DECAY_CHALLENGE            = 0.50              // per upheld challenge
Q_DECAY_AUDIT                = 0.70              // per failed audit
Q_RECOVERY_PER_SETTLEMENT    = 1.0001            // multiplicative recovery cap 1.0
Q_MIN_FOR_EMISSION           = 0.10              // below: no emission share

// BUYBACK
BUYBACK_THRESHOLD_USDC       = 100
BUYBACK_SLIPPAGE_BPS         = 100               // 1.0%
BUYBACK_POOL_PCT             = 50                // 0.5% of liquidity per tx
BURN_ADDRESS                 = "1nc1nerator11111111111111111111111111111111"

// EARLY RAMP (first 72 hours)
RAMP_0H_TO_24H_CAP_PCT       = 10                // 10% of normal rate
RAMP_24H_TO_72H_CAP_PCT      = 50                // 50% of normal rate
RAMP_72H_PLUS                = 100               // full rate

// VESTING (for accumulated CLAF rewards)
REWARD_VESTING_DAYS          = 180
REWARD_VESTING_LINEAR        = true

// PROGRAM IDs (placeholders; set at Genesis)
PROVIDER_REGISTRY_PROGRAM    = "..." // Solana program ID
SETTLEMENT_ESCROW_PROGRAM    = "..."
```

```
TOKEN_METERING_PROGRAM       = "..."
TREASURY_PROGRAM             = "..."
CLAF_MINT                    = "..."
```

The above constants will be replaced with their final values in the Genesis-deployment manifest, published at Genesis to `github.com/clawfarm-protocol/genesis` together with the bytecode hashes of the four programs.

⁂

Appendix B. Pseudocode for Core Programs

We give pseudocode for the four programs' instructions, abbreviated for readability. Production code is in the reference implementation.

B.1 Provider Registry

```
fn register_provider(ctx, offerings: Vec<Offering>) {
    require!(!ctx.accounts.provider_state.initialized);
    require!(offerings.len() > 0 && offerings.len() <= 64);

    // Transfer 100 USDC bond from provider wallet to bond PDA
    transfer_usdc(
        from: ctx.accounts.provider_wallet,
        to:   ctx.accounts.bond_pda,
        amount: PROVIDER_BOND_USDC,
    );

    // Initialize provider state
    ctx.accounts.provider_state = ProviderState {
        wallet:       ctx.accounts.provider_wallet.key,
        bond_amount:  PROVIDER_BOND_USDC,
        offerings:    offerings,
        q_factor:     Q_INITIAL,
        registered_at: clock::now(),
        active:       true,
    };

    emit!(ProviderRegistered { provider: ctx.accounts.provider_wallet.key });
}

fn update_offering(ctx, new_offerings: Vec<Offering>) {
    require!(ctx.accounts.provider_state.active);
    require!(ctx.accounts.signer == ctx.accounts.provider_state.wallet);
    ctx.accounts.provider_state.offerings = new_offerings;
}

fn withdraw_bond(ctx) {
    require!(ctx.accounts.signer == ctx.accounts.provider_state.wallet);
    require!(!has_open_challenges(ctx.accounts.provider_state));
    require!(clock::now() - last_settlement(ctx.accounts.provider_state)
             >= BOND_COOLDOWN_DAYS * 86400);

    transfer_usdc(
        from: ctx.accounts.bond_pda,
        to:   ctx.accounts.provider_wallet,
        amount: ctx.accounts.provider_state.bond_amount,
    );
    ctx.accounts.provider_state.active = false;
}
```

B.2 Settlement Escrow (excerpt: settle_session)

```
fn settle_session(ctx, proof: UsageProof) {
    // Verify proof signatures
    verify_ed25519(proof.user, proof.body(), proof.user_signature);
    verify_ed25519(proof.provider, proof.body(), proof.provider_signature);

    // Verify session exists and is open
    let session = ctx.accounts.session_state;
    require!(session.user == proof.user);
    require!(session.provider == proof.provider);
    require!(session.status == SessionStatus::Open);
    require!(clock::now() < session.opened_at + SESSION_TIMEOUT_HOURS * 3600);

    // Compute settlement amount in USDC
    let total_tokens = proof.input_tokens + proof.output_tokens;
    let usdc_amount = total_tokens * proof.declared_price / 1_000_000;
    require!(usdc_amount <= session.reserved_amount);

    // 97% to provider, 3% to treasury
    let provider_amount = usdc_amount * PROVIDER_SHARE_BPS / 10_000;
    let protocol_amount = usdc_amount - provider_amount;

    transfer_usdc(
        from: ctx.accounts.user_escrow_pda,
        to:   provider_wallet_of(proof.provider),
        amount: provider_amount,
    );
    transfer_usdc(
        from: ctx.accounts.user_escrow_pda,
        to:   ctx.accounts.treasury_collection_pda,
        amount: protocol_amount,
    );

    // Return over-reservation
    let refund = session.reserved_amount - usdc_amount;
    if refund > 0 {
        unencumber(ctx.accounts.user_escrow_pda, refund);
    }

    // Update epoch statistics for reward weight computation
    update_epoch_stats(proof.provider, total_tokens, proof.declared_price);

    session.status = SessionStatus::Settled;
    emit!(SessionSettled { /* ... */ });
}
```

B.3 Treasury (excerpt: execute_buyback)

```
fn execute_buyback(ctx) {
    let balance = balance_of(ctx.accounts.treasury_collection_pda);
    require!(balance >= BUYBACK_THRESHOLD_USDC);

    let pool_depth = jupiter::query_pool_depth(USDC_CLAF_POOL);
    let max_swap = pool_depth * BUYBACK_POOL_PCT / 10_000;
    let swap_amount = min(balance, max_swap);

    // Cross-program invocation to Jupiter
    let claf_received = jupiter::swap_exact_in(
        from_mint:   USDC_MINT,
        to_mint:     CLAF_MINT,
        amount_in:   swap_amount,
        min_amount_out: estimate_out(swap_amount) * (10_000 - BUYBACK_SLIPPAGE_BPS) /
10_000,
        from_account: ctx.accounts.treasury_collection_pda,
        to_account:   ctx.accounts.buyback_holding_pda,
    );

    // Transfer all received CLAF to burn address
    transfer_claf(
        from: ctx.accounts.buyback_holding_pda,
        to:   BURN_ADDRESS,
        amount: claf_received,
    );

    emit!(BuybackExecuted { usdc_in: swap_amount, claf_burned: claf_received });
}
```

⁂

Appendix C. Notation Summary

Symbol	Meaning
T_i	Token volume served by provider i in an epoch
P_i	Volume-weighted price charged by provider i in an epoch
P_{avg}	Network-wide volume-weighted average price in an epoch
Q_i	Quality factor of provider i, bounded in [0, 1]
W_i	Provider Pool reward weight of provider i = Ti · (Pavg/Pi) · Qi

c_i	Marginal USDC cost of provider i per million tokens
E_t	CLAF emission in epoch t (fixed by schedule)
B_t	CLAF burned in epoch t via buyback
U_t	USDC volume settled in epoch t
$P_{\text{CLAF},t}$	CLAF price in USDC at epoch t
N	Number of active providers
D	Exogenous user demand
Π_i	Provider i's total expected payoff (USDC + CLAF-discounted)

⁂

Appendix D. Solana Account Schemas

We give the account layouts for the protocol's principal state accounts. Field sizes are in bytes; the schemas are Anchor-compatible.

```
ProviderState {
    discriminator:       [u8; 8],         // Anchor account discriminator
    wallet:              Pubkey,          // 32
    bond_amount:         u64,             // 8 (lamports of USDC, 6 decimals)
    offerings_count:     u8,              // 1
    offerings:           [Offering; 64],  // 64 * 48
    q_factor_x1000:      u16,             // 2 (Q × 1000, integer)
    cumulative_tokens:   u64,             // 8
    last_epoch_index:    u64,             // 8
    last_settlement_ts:  i64,             // 8
    registered_at:       i64,             // 8
    active:              bool,            // 1
    _reserved:           [u8; 32],        // 32
}
// Total: ~3,200 bytes

Offering {
    model_id:            [u8; 32],        // 32
    price_per_mtok:      u64,             // 8 (USDC lamports per MTok)
    declared_quality:    u16,             // 2
    max_concurrent:      u32,             // 4
    _reserved:           [u8; 2],         // 2
}
// Total: 48 bytes

SessionState {
    discriminator:       [u8; 8],
    user:                Pubkey,          // 32
    provider:            Pubkey,          // 32
    model_id:            [u8; 32],        // 32
    declared_price:      u64,             // 8
    reserved_amount:     u64,             // 8
    opened_at:           i64,             // 8
    status:              u8,              // 1: Open / Settled / Expired / Challenged
    _reserved:           [u8; 7],         // 7
}
// Total: 144 bytes

EscrowState {
    discriminator:       [u8; 8],
    user:                Pubkey,          // 32
    balance:             u64,             // 8 (USDC lamports)
    encumbered:          u64,             // 8 (USDC reserved against open sessions)
    deposit_count:       u32,             // 4
    last_deposit_ts:     i64,             // 8
    _reserved:           [u8; 16],        // 16
}
// Total: 84 bytes

TreasuryState {
    discriminator:       [u8; 8],
    cumulative_fees:     u64,             // 8
    cumulative_burns:    u64,             // 8 (CLAF burned)
    last_buyback_ts:     i64,             // 8
    last_emission_ts:    i64,             // 8
    cumulative_emitted:  u64,             // 8
```

```
    _reserved:           [u8; 32],        // 32
}
// Total: 80 bytes

EpochStats {
    discriminator:       [u8; 8],
    epoch_index:         u64,             // 8
    total_tokens:        u64,             // 8
    total_volume_usdc:   u64,             // 8
    avg_price:           u64,             // 8 (volume-weighted)
    provider_count:      u32,             // 4
    finalized:           bool,            // 1
    _reserved:           [u8; 19],        // 19
}
// Total: 88 bytes
```

Account allocation per program follows Solana's rent-exempt minimums; the protocol's rent burden at scale is bounded by the linear growth of EpochStats and ProviderState accounts, with all other account types being per-session or per-user and naturally bounded.

⁂

END OF VERSION 2.0 · GENESIS DRAFT · CLAWFARM PROTOCOL