Hellas: Economic Actors, Value Flows, and Incentive Conditions

CryptoEconLab

February 23, 2026

Abstract

This document specifies the economic structure of the Hellas protocol. It defines the roles of validators, clients, and providers, the value flows induced by state channel lifecycle events, and the game-theoretic conditions that ensure honest execution and credible disputes. The analysis establishes when the protocol's economic incentives make cheating unprofitable for rational providers and when clients find it worthwhile to challenge incorrect results. Unspecified parameters are represented as explicit variables.

Contents

1 Economic Actors and Their Roles

The protocol has three economically meaningful actor classes: validators, clients, and providers. A *state channel* is a bilateral off-chain interaction between one client and one provider, with escrow and collateral locked on the Hellas Layer 1. The Layer 1 finalizes channel opening and closing, and adjudicates disputes by verifying succinct proofs.

1.1 Validators

Validators operate the Hellas Layer 1 consensus (Minimmit protocol). Validators propose and finalize blocks, process state channel opening and closing transactions, and verify succinct dispute proofs submitted on-chain. When a proof is valid, the protocol deterministically executes the corresponding penalty, including slashing the challenged provider stake when applicable.

Economic Requirements. To participate, validators must stake Hellas tokens. Let S_V denote a validator's stake. Let r denote the per-unit-time opportunity cost of locked capital, expressed in tokens per token per unit time. Let c_{infra} denote validator infrastructure and operational cost, expressed in tokens per unit time.

Revenue Sources. Validator revenue sources are (i) protocol fees paid on channel lifecycle transactions (Section 4) and (ii) token emissions determined by an emission schedule. Dispute verification is compensated only through standard transaction fees, not through a share of slashed provider stake.

Design Choice TBD

The validator emission schedule $E_V(t)$ is not yet defined. This parameter determines the extent to which validator security is funded by inflation versus usage fees.

1.2 Clients

Clients request AI inference and open state channels with providers. A client locks escrow on-chain, pays protocol fees, and receives an off-chain result from the provider. If the client believes the provider returned an incorrect result, the client can verify correctness through a *safe fallback* mechanism and, conditional on discovering fraud, submit a succinct proof on-chain to trigger slashing.

Economic Requirements. A client locks escrow E_C to open a channel. Let $P_{\max}$ denote the maximum on-chain payment that can be made to the provider under the channel rules, and let P_{set} denote the payment realized at settlement. The protocol requires $E_C \geq P_{\max}$ and enforces $0 \leq P_{\text{set}} \leq P_{\max}$.

Key Insight

The client is solely responsible for detecting and proving incorrect execution. No third party can observe the off-chain transcript. This design preserves privacy but concentrates verification incentives on the client. The economic viability of this design depends on the *Fraud Game* (Section 6), which ensures that clients have sufficient incentive to challenge incorrect results and that providers face credible punishment.

Note

Define the escrow buffer $\Delta_C := E_C - P_{\max} \geq 0$. The buffer can be set to zero for fixed-price jobs. A positive buffer is useful when payment is metered or when worst-case payment must be bounded at channel opening.

1.3 Providers

Providers supply compute and execute AI workloads off-chain under channel-defined rules. A provider locks stake as collateral against incorrect execution and returns results to clients off-chain.

Economic Requirements. Providers lock stake S_P that is slashable upon a valid on-chain proof of incorrect execution. The stake is locked per channel. Let τ denote the channel duration, measured in the same unit as r. The provider incurs an opportunity cost $rS_P\tau$.

Stake Flexibility. Providers can choose S_P job by job. The stake S_P is publicly observable on-chain at channel opening and can be treated as an explicit, priced security level.

Note

A higher stake S_P increases the provider's cost of capital and therefore tends to increase the equilibrium price charged for service. This creates a market link between security level and payment, but it is not an identity and should not be assumed without an explicit incentive argument.

2 State Channel Lifecycle

This section describes the lifecycle of a single client-provider channel. The channel is opened and closed on-chain, while execution occurs off-chain. The on-chain contract enforces escrow and stake accounting, and defines a dispute window during which a valid proof can trigger slashing. The dispute mechanism is the protocol's core security primitive: it creates economic consequences for incorrect execution that, when properly calibrated, make honest behavior the rational choice for providers.

2.1 Happy Path: Successful Execution

1. **Off-chain Agreement.** The client and provider agree off-chain on a job specification and pricing.
2. **Channel Opening.** The on-chain opening transaction locks client escrow E_C and provider stake S_P, and records a commitment to the job parameters needed for later verification.
3. **Off-chain Execution.** The parties interact off-chain, with the provider executing the specified deterministic workload and returning results to the client.
4. **Cooperative Channel Closing.** If the client accepts the result, the parties cooperatively close the channel by signing a final state that specifies the settlement payment P_{set}. The closing transaction releases P_{set} to the provider, returns $E_C - P_{\text{set}}$ to the client, and unlocks S_P.

2.2 Unhappy Path: Dispute Resolution

A dispute arises when the client does not accept the provider's output and the parties cannot produce a cooperative close. The protocol enables the client to establish incorrect execution through a *safe fallback* mechanism and then produce a succinct proof whose verification is cheap for validators.

Safe Fallback. A safe fallback is any method that allows the client to compute the correct result for a job. The simplest implementation is for the client to re-execute the job on their own infrastructure or to purchase re-execution from an independent provider. More generally, trusted fallback providers can supply verification as a service. Let C_{safe} denote the total cost of the safe fallback computation, including any fees, infrastructure overhead, or service charges.

Dispute Flow. If the client's safe fallback reveals that the original provider returned an incorrect result, the client generates a fraud proof at cost c_{proof}, submits it on-chain along with a challenge bond B_C, and pays transaction costs c_{tx}. Validators verify the proof. If the proof is valid, the protocol slashes the provider's stake S_P. A fraction β of the slashed stake is paid to the client, and a fraction λ of the escrowed payment P_{set} is rerouted to the client. The bond B_C is returned when the proof is valid and forfeited when the proof is invalid.

Note

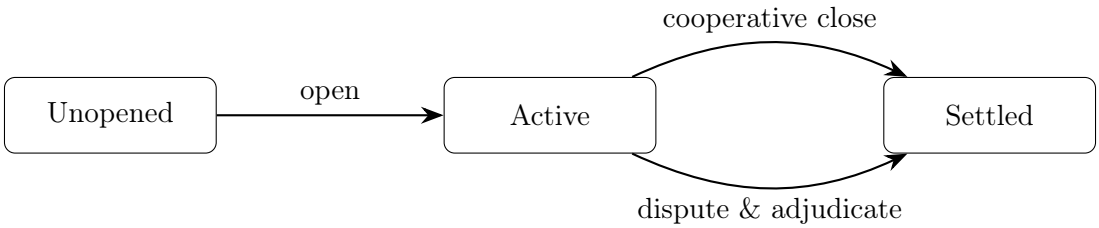
Slashed stake is used to compensate the harmed client, not to fund validators or token value accrual. A slashing event corresponds to a failed service outcome. The economic intent is that protocol value accrual should be driven by successful usage fees, not by slashing events.

Anti-Griefing. The challenge bond B_C serves as an anti-griefing mechanism. If a client submits an invalid proof (either maliciously or in error), the bond is forfeited to compensate for the system overhead. When the enforcement mechanism is reliable, meaning when the probability that valid proof is accepted and enforced is high, the bond can be small because rational clients only challenge when they have strong evidence of fraud. When enforcement is less reliable, the bond must be larger to discourage frivolous or speculative challenges.

Key Insight

The dispute mechanism is the foundation of Hellas's trustless guarantees. Without a credible dispute process, clients cannot verify that providers are honest, and rational clients will not join the network. The *Fraud Game* analysis in Section 6 establishes the precise conditions under which disputes are credible and deterrence is effective.

2.3 State Diagram



3 Parameters and Notation

This section defines the variables used in the economic and incentive analysis. All token quantities are denominated in Hellas tokens unless otherwise stated.

3.1 Actor-Specific Parameters

Symbol	Description	Set By
S_V	Validator stake	Validator
S_P	Provider stake (per channel)	Provider
E_C	Client escrow (per channel)	Client
$P_{\max}$	Maximum payable amount under channel rules	Bilateral (auction)
P_{set}	Settlement payment released to provider	Channel state
B_C	Client challenge bond	Protocol
c_H	Provider cost of honest execution	Exogenous
c_F	Provider cost under cheating strategy	Exogenous
C_{safe}	Cost of safe fallback computation	Market / job type
c_{proof}	Cost of fraud proof generation	Exogenous
c_{tx}	On-chain transaction cost for disputing	Network conditions
L	Client loss from accepting an incorrect result	Job dependent
r	Opportunity cost rate of locked capital	Market
τ	Channel duration	Endogenous

Table 1: Actor-specific economic parameters

The parameter L represents the job-specific loss that a client incurs if they accept an incorrect result without detecting the fraud. This loss varies by application: for a low-stakes inference query, L may be negligible; for a financial model or safety-critical computation, L may far exceed the payment P_{set}. The magnitude of L influences how aggressively clients audit results, as shown in Section 6.

Note

The bond B_C can be separate from escrow or can be carved out of escrow, subject to $B_C \leq E_C$. Using escrow for the bond increases capital efficiency but weakens the bond as an anti-spam mechanism if escrow is always present for unrelated reasons.

3.2 Protocol Parameters

Symbol	Description	Status
α	Ad valorem fee rate on settled payment P_{set}	TBD
ϕ	Fraction of fees burned	TBD
β	Fraction of slashed stake paid to client	$\beta \in (0, 1]$, default $\beta = 1$
λ	Fraction of P_{set} rerouted to client on success	$\lambda \in [0, 1]$, default $\lambda = 1$
p_w	Probability a valid proof is accepted and enforced	Protocol environment
$E_V(t)$	Validator emission schedule	TBD
T	Dispute timeout (blocks)	TBD

Table 2: Protocol-level parameters

The parameter p_w captures the reliability of the enforcement mechanism. In an ideal protocol, $p_w = 1$, meaning that every valid fraud proof is accepted and leads to slashing of the fraudulent provider. In practice, $p_w < 1$ may arise from censorship risk, liveness failures, deadline misses, or verifier bugs. Lower p_w weakens deterrence and requires higher stakes to compensate.

4 Fee Structure

Hellas charges fees to fund validator security and to capture a small fraction of value created by successful settlement. A fee component that is proportional to *settled value* avoids taxing unused escrow buffers.

Definition 1 (Proposed Channel Fees)**.**

$$F_{val} = \alpha \cdot P_{set} \tag{1}$$

$$F_{total} = F_{open,net} + F_{close,net} + F_{val} \tag{2}$$

Here $F_{\text{open,net}}$ and $F_{\text{close,net}}$ are the standard network fees for the opening and closing transactions, which may be implemented as a base fee plus a priority fee. The ad valorem component F_{val} is assessed at settlement when P_{set} is known.

4.1 Fee Distribution

For the ad valorem fee F_{val}, define the distribution rule

$$\text{Burned} = \phi \cdot F_{\text{val}}, \tag{3}$$

$$\text{To block producer} = (1 - \phi) \cdot F_{\text{val}}. \tag{4}$$

This is compatible with an EIP-1559 style mechanism in which the base fee is burned and the priority fee is paid to the block producer.

Note

An ad valorem fee on P_{set} can be justified even when the marginal on-chain cost of settlement is not proportional to P_{set}. The mechanism is a protocol pricing rule that targets value created by off-chain execution and on-chain enforceability. Users who prefer to avoid the fee can transact fully off-chain, but then they forgo on-chain settlement and slashing enforceability.

Open Question

If fees are charged on P_{set} rather than E_C, the relevant question becomes whether *settled payment* is a proxy for security level. A client can only obtain "cheap security" if a high-stake provider is willing to accept a low P_{set}. That is feasible only as a subsidy strategy by the provider. If such subsidization is rare, then P_{set} is an informative proxy for S_P through the provider's cost of capital. If client and provider attempt to move value off-channel to reduce P_{set}, they reduce the amount that is enforceable by the channel and thus reduce the economic value protected by slashing.

5 Value Flows

This section records per-channel value flows. Opportunity costs of locked funds are modeled explicitly where they affect incentives.

5.1 Validator Value Flow

$$\boxed{\begin{aligned}\text{Validator Revenue} &= \underbrace{(1-\phi)\,F_{\text{val}}}_{\text{ad valorem fees}} + \underbrace{F_{\text{open,net}} + F_{\text{close,net}}}_{\text{standard tx fees}} + \underbrace{E_V(t)}_{\text{emissions}} \\ \text{Validator Cost} &= \underbrace{r \cdot S_V}_{\text{stake opportunity cost}} + \underbrace{c_{\text{infra}}}_{\text{infrastructure}}\end{aligned}} \tag{5}$$

This expression is written per channel close event. For throughput-level analysis, sum the fee terms over events per unit time.

Note

Validators do not receive any share of slashed provider stake. Slashing is modeled as a client compensation transfer.

5.2 Client Value Flow

5.2.1 Happy Path (No Dispute)

$$\boxed{\text{Client Outlay} = P_{\text{set}} + F_{\text{open,net}} + F_{\text{close,net}} + \alpha P_{\text{set}}.} \tag{6}$$

The escrow E_C is working capital that is returned net of P_{set} at settlement. If the opportunity cost of locked escrow is material, an additional term $rE_C\tau$ should be included in client costs.

5.2.2 Dispute: Client Wins

When the client successfully proves fraud, the value flow reflects both the cost of verification and the rewards from slashing. Define the total dispute cost as

$$C_{\text{disp}} := C_{\text{safe}} + c_{\text{proof}} + c_{\text{tx}}. \tag{7}$$

This is the full cost of safe fallback computation, proof generation, and on-chain submission.

If the proof is valid and enforcement succeeds (probability p_w), the client receives reward

$$R := \beta S_P + \lambda P_{\text{set}}, \tag{8}$$

where βS_P is the share of slashed stake and λP_{set} is the routed payment. The bond B_C is returned. If enforcement fails (probability $1 - p_w$), the client forfeits the bond B_C and receives nothing.

$$\boxed{\text{Client Net (Dispute)} = p_w \cdot R - C_{\text{disp}} - (1-p_w) \cdot B_C.} \tag{9}$$

This expression determines whether disputing is privately profitable for the client, which is essential for the Fraud Game analysis in Section 6.

5.3 Provider Value Flow

5.3.1 Honest Execution

$$\boxed{\pi_H = P_{\text{set}} - c_H - rS_P\tau.} \tag{10}$$

5.3.2 Cheating and Detected

$$\boxed{\pi_D = -S_P - rS_P\tau.} \tag{11}$$

Under detection, the provider forfeits stake and does not receive P_{set}.

5.3.3 Cheating and Not Detected

$$\boxed{\pi_U = P_{\text{set}} - c_F - rS_P\tau.} \tag{12}$$

The stake S_P is returned when cheating is not detected. It is not an additional gain.

6 The Fraud Game

The Fraud Game is the game-theoretic foundation of Hellas's security. It determines whether rational, self-interested actors will behave honestly. If the Fraud Game is not properly calibrated, providers can cheat with impunity, clients cannot trust results, and the network fails to deliver trustless AI computation. This section presents the main results that characterize when the Fraud Game works.

The analysis separates two components:

1. **Enforcement**: Conditional on fraud being discovered, is it privately optimal for the client to submit a dispute?
2. **Detection**: When clients cannot immediately verify correctness, how often do they audit, and how does this affect provider behavior?

6.1 Enforcement: When Disputes Are Credible

For the Fraud Game to deter cheating, clients must actually submit disputes when they discover fraud. If disputing is not privately profitable, rational clients will not dispute, and providers can cheat without consequence.

Dispute Profitability. A client who has discovered fraud will submit a dispute if and only if the expected reward exceeds the expected cost:

$$p_w \cdot R - C_{\text{disp}} - (1 - p_w) \cdot B_C > 0, \tag{13}$$

where $R = \beta S_P + \lambda P_{\text{set}}$ is the reward on success, $C_{\text{disp}} = C_{\text{safe}} + c_{\text{proof}} + c_{\text{tx}}$ is the dispute cost, and B_C is forfeited on failure.

Minimum Viable Stake. The dispute profitability condition can be rearranged to yield the minimum provider stake that makes disputing worthwhile:

$$\boxed{S_P^{\min} = \max\left(0,\ \frac{C_{\text{disp}} + (1 - p_w)B_C - p_w\lambda P_{\text{set}}}{p_w\beta}\right).} \tag{14}$$

This is a central result. If $S_P < S_P^{\min}$, then even when fraud is discovered, disputing is not profitable and the client will not challenge. In this *impunity region*, providers face no economic consequence for cheating.

Key Insight

The minimum viable stake $S_P^{\min}$ depends on three protocol levers:

- **Reward share** β: Increasing the fraction of slashed stake paid to the challenger reduces $S_P^{\min}$.
- **Payment routing** λ: Routing more of the escrowed payment to the challenger on success reduces $S_P^{\min}$.
- **Enforcement reliability** p_w: Higher reliability reduces $S_P^{\min}$ by making the expected reward more certain.

6.2 Detection: The Inspection Game

Even when disputes are credible (i.e., $S_P \geq S_P^{\min}$), fraud may persist if clients do not audit results. Auditing is costly: the client must pay C_{safe} to verify correctness. If auditing is too expensive relative to the potential loss from accepting a wrong result, clients will not audit, and providers can cheat undetected.

The interaction between provider cheating and client auditing is modeled as an *inspection game*: i.e. a simultaneous-move game where the provider chooses whether to cheat and the client chooses whether to audit, without observing the other's action.

Mixed Equilibrium. When neither pure honesty nor pure cheating dominates, the game has a mixed-strategy equilibrium where both parties randomize. Let q denote the probability the provider cheats and v denote the probability the client audits. The equilibrium values are:

$$\boxed{v^* = \frac{c_H - c_F}{P_{\text{set}} + S_P}} \tag{15}$$

$$\boxed{q^* = \frac{C_{\text{safe}}}{L + \Delta}} \tag{16}$$

where $\Delta = p_w(\beta S_P + \lambda P_{\text{set}}) - (c_{\text{proof}} + c_{\text{tx}}) - (1 - p_w)B_C$ is the expected net surplus from disputing conditional on finding fraud.

Interpretation. The equilibrium audit rate v^* is the detection probability that makes providers indifferent between cheating and honest execution. It equals the cost savings from cheating $(c_H - c_F)$ divided by the total at-risk amount $(P_{\text{set}} + S_P)$. Larger stakes and payments reduce the required audit rate.

The equilibrium cheating rate q^* is the fraud probability that makes clients indifferent between auditing and accepting. It equals the audit cost (C_{safe}) divided by the benefit of auditing when fraud occurs $(L + \Delta)$. Larger job-specific loss L and larger dispute surplus Δ reduce equilibrium cheating.

Key Insight

The equilibrium cheating rate q^* decreases when:

- Provider stake S_P increases (raises Δ through βS_P)
- Job-specific loss L increases (raises client incentive to audit)
- Safe fallback cost C_{safe} decreases (lowers audit threshold)
- Enforcement reliability p_w increases (raises Δ)

These comparative statics guide protocol design: to reduce fraud, increase stakes, reduce verification costs, and improve enforcement reliability.

6.3 Provider Incentive Compatibility

Combining enforcement and detection, we can state the condition under which providers prefer honest execution. Let p_d denote the probability that cheating is detected and punished. In the dispute subgame where $S_P \geq S_P^{\min}$, we have $p_d = p_w \cdot v$ where v is the audit probability.

Condition 1 (Provider Honesty)**.** *A provider prefers honest execution over cheating if and only if*

$$\boxed{p_d \geq \frac{c_H - c_F}{P_{set} + S_P}.} \tag{17}$$

When $c_F \approx 0$ (cheating is nearly costless), this simplifies to $p_d \geq \frac{c_H}{P_{set}+S_P}$.

This condition shows that deterrence requires the detection probability p_d to exceed a threshold determined by the cost savings from cheating and the total at-risk amount. The protocol can satisfy this condition by ensuring sufficient stakes, sufficient audit rates, or both.

6.4 Boundary Cases and Failure Modes

The Fraud Game can fail in several ways:

Impunity Region. If $S_P < S_P^{\min}$, disputes are not profitable and providers face no punishment. The effective detection probability p_d collapses to zero regardless of client auditing behavior.

No Auditing Equilibrium. If $C_{\text{safe}} \geq L + \Delta$, auditing is never worthwhile for clients. The equilibrium is $v^* = 0$ and $q^* = 1$: providers always cheat and clients never audit. This occurs when verification is too expensive relative to the combined benefit of avoiding loss and obtaining dispute rewards.

Low-Value Jobs. For jobs where P_{set} is small, the minimum viable stake $S_P^{\min}$ may be large relative to the payment, making the job uneconomical for providers. Additionally, if L is small, clients have little incentive to audit. Low-value jobs may require batching, minimum payment floors, or explicit "low-security tier" labeling.

Open Question

A concrete policy is required for low-value jobs. Options include: impose a minimum P_{set} or minimum S_P; batch jobs so that payments scale while proof overhead amortizes; or expose an explicit security tier where low-stake jobs are labeled as providing weaker deterrence.

7 Token Economics and Value Accrual

Token demand arises from stake and working capital requirements. Validators lock stake. Providers lock per-channel collateral. Clients lock escrow and, during disputes, challenge bonds. These locks are protocol-mandated and scale with usage and chosen security levels.

7.1 Token Supply Dynamics

Definition 2 (Circulating Supply)**.** *At any time t:*

$$\mathit{Circulating}(t) = \mathit{Total\ Supply} - \mathit{Locked}(t) - \mathit{Burned}(t), \tag{18}$$

where

$$\mathit{Locked}(t) = \sum_i S_{V,i} + \sum_{\mathit{active\ channels}} (S_P + E_C) + \sum_{\mathit{active\ disputes}} B_C, \tag{19}$$

$$\mathit{Burned}(t) = \int_0^t \phi \cdot F_{val}(\tau)\, d\tau. \tag{20}$$

7.2 Lockup Effect

Let λ_{open} be the channel opening rate and let $\bar{\tau}$ be the average channel duration. In steady state, the expected quantity of tokens locked in channels satisfies

$$\text{Locked in Channels} \approx \lambda_{\text{open}} \bar{\tau}\, \mathbb{E}[S_P + E_C]. \tag{21}$$

Higher usage and higher demanded security levels increase working-capital lockup.

7.3 Value Accrual Mechanisms

Protocol value accrual is usage-driven. The primary accrual mechanism is fee burning through ϕF_{val}. Lockup demand is a working-capital effect that can increase token demand as usage scales. Slashing is treated as a compensation transfer and is not a designed accrual mechanism.

Note

In an equilibrium with well-calibrated incentives, slashing is rare. Sustainable protocol economics should therefore be robust to low slashing frequency.

Design Choice TBD

The joint choice of $(\alpha, \phi, E_V(\cdot))$ determines whether validator security is primarily funded by usage fees or emissions and determines the net inflation or deflation profile. Calibration requires an assumed usage path and target validator security budget.

8 Summary and Open Questions

8.1 Current Design Commitments

The current design commits to the following:

- **No separate attester role**: Disputes are initiated by clients via safe fallback verification. This preserves privacy but concentrates verification incentives on clients.
- **Client-funded verification**: Proof generation is performed client-side and is intended to be substantially cheaper than re-executing the full workload.
- **Usage-based fees**: Fees are charged as a small fraction of settled payment and are intended to fund validators through usage, not through slashing.
- **Slashing as compensation**: Slashed stake goes to the harmed client ($\beta = 1$ by default), not to validators or token burning.
- **Deterministic execution**: Provider execution is assumed deterministic given the channel-defined job specification, enabling unambiguous fraud proofs.

8.2 Parameters Requiring Specification

The remaining parameters are α, ϕ, $E_V(t)$, T, and B_C. The piece sets $\beta = 1$ and $\lambda = 1$ by default to maximize client incentives to dispute and to align dispute rewards with dispute costs.

9 Conclusion

This document specifies the economic architecture of the Hellas protocol for trustless AI computation. The key insight is that trustlessness emerges from economic incentives, not from cryptographic verification of every computation. The Fraud Game creates conditions under which rational providers prefer honest execution because the expected cost of cheating exceeds the benefit.

Value Flow Summary. Value flows through Hellas as follows:

- **Clients** pay providers for computation (P_{set}) and pay the protocol fees (αP_{set}). They lock escrow as working capital and, if they dispute, incur verification costs (C_{disp}) but can recover slashed stake (βS_P) and routed payment (λP_{set}).
- **Providers** earn payment minus execution cost ($P_{\text{set}} - c_H$), net of capital costs ($rS_P\tau$). Their stake is at risk: if caught cheating, they lose S_P and P_{set}.
- **Validators** earn fees from channel lifecycle transactions and token emissions. They do not share in slashing rewards.
- **Token holders** benefit from fee burning (ϕF_{val}), which reduces circulating supply, and from lockup demand as usage scales.

Fraud Game Summary. The Fraud Game analysis yields two key constraints:

1. **Credible disputes**: Provider stake must exceed $S_P^{\min}$ to make disputing profitable for clients. Below this threshold, providers operate with impunity.
2. **Sufficient deterrence**: The effective detection probability p_d must exceed $\frac{c_H - c_F}{P_{\text{set}} + S_P}$ to make honesty preferable to cheating.

The equilibrium fraud rate $q^* = \frac{C_{\text{safe}}}{L+\Delta}$ shows how protocol parameters affect security: lower verification costs, higher stakes, and larger job-specific losses all reduce cheating.

Design Implications. For Hellas to succeed as a trustless network:

- Staked amounts should be high enough that $S_P \geq S_P^{\min}$ for each job class.
- Verification costs (C_{safe}, c_{proof}) should be minimized through efficient proof systems and competitive fallback provider markets.
- High-value jobs (large L) naturally receive more auditing and thus stronger security guarantees.
- The protocol should be robust to low slashing frequency, since well-calibrated incentives should make slashing rare.

The economic structure is designed so that the network becomes more secure and more valuable as usage grows: higher throughput increases fee revenue and lockup demand. A more widely used system creates larger economic opportunities for trusted fallback providers, fostering a competitive market for verification services that lowers C_{safe} and reduces the equilibrium fraud rate. This creates a flywheel where demand for trustless AI compute reinforces the economic foundations that make trustlessness possible.