

Inference-Based Decentralised Value: TensorCash

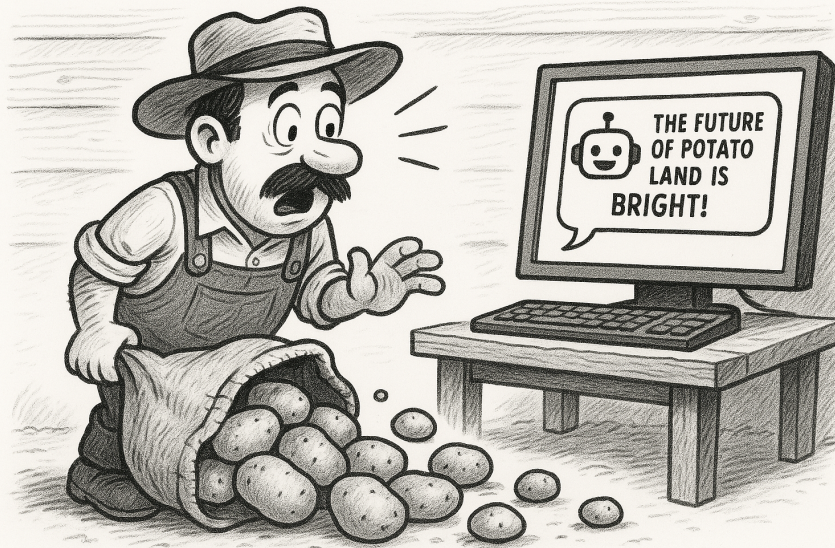
Imosuke Takakuni¹

1 April 2025

ABSTRACT

TensorCash is a protocol for making AI inference efficiently verifiable. It lets an inference provider attach a proof-of-inference transcript to any AI model run. Users, aggregators and validators can check that a model was executed under declared conditions without trusting the provider. The same proof-of-inference primitive can then be used as proof of work in the sense introduced by Nakamoto [2008]: TensorCash introduces a Bitcoin-derived blockchain and native token whose block production is secured by useful inference. AI providers can secure a cryptocurrency while serving commercially valuable tasks, converting inference into decentralised economic value.

Introduction: The Parable of Potato-Land



In Potato-Land, potatoes were the trusted currency—easily traded and widely accepted. Sure, there were minor issues: each farm had its own potato variety, and some potatoes were more perishable than others. Still, Potato-Landers adapted and got used to these quirks.

Life was simple until AI technology emerged. It sparked immense excitement and demand. Big companies eagerly spent vast stockpiles of potatoes building machines to harness this revolutionary technology. Potato-Landers enthusiastically traded their hard-earned potatoes for access, craving more and more of its benefits.

1. "Imosuke Takakuni" is a pseudonym. Contact: takakuni@tensorcash.org

Then scientists made a clever adjustment. Suddenly, any company providing AI services could also effortlessly run a Proof-of-Work (PoW) algorithm in the background, earning a new cryptocurrency called "TensorCash". Initially, skepticism surrounded this novel currency. But forward-thinking AI providers quickly realized this required minimal additional effort and yielded generous mining rewards.

Gradually, Potato-Landers discovered that transactions using TensorCash were cheaper, easier, more secure, and far more practical compared to storing or transporting potatoes. Unlike potatoes, TensorCash was not perishable and wasn't impacted by the variability of individual farm harvests, or the whim of their ruling farmers.

Soon, large service providers preferred transactions in TensorCash, prompting hardware manufacturers to accept TensorCash payments directly from their significant customers. Eventually, salaries and everyday transactions within tech industries shifted toward TensorCash. Market incentives—initial mining rewards, transaction convenience, and inherent practicality—organically propelled TensorCash to dominance. While potatoes remained valuable commodities, TensorCash's advantages as a stable, efficient, and practical medium of exchange ultimately made it Potato-Land's preferred currency.

by: ChatGPT 4o, March 2025.

Back on Earth

Generative AI has been heralded as the new Industrial Revolution, its promise spanning from utopian visions of a work-free golden age to dystopian fears of extinction at the hands of superintelligent machines. Amid hype and uncertainty, few things are clear: demand for AI tools is durable and growing, models are being competitively developed, and vast amounts of computational infrastructure are being provisioned.

This capacity is not being built only to train ever-larger models. It is also being built to support the massive and ongoing demand for AI inference: the real-time deployment and usage of these models. In generative AI, inference carries particular weight. Generating a single word, audio segment, video frame, or image region can require billions or trillions of floating-point operations on GPUs or hardware accelerators. At global scale, serving models can become more resource-intensive and expensive than training them.

The same period also saw the rise of decentralised finance and cryptographic settlement. From Satoshi Nakamoto's pioneering vision, a global system emerged to store and transact value pseudonymously, directly, and with finality. Bitcoin introduced a peer-to-peer ledger in which participants agree on transactions by extending a "block" chain with the greatest accumulated proof-of-work effort [Nakamoto, 2008]. This attracted trillions of dollars in capitalisation, raised existential questions for fiat currencies, and redefined how we think about financial exchange and settlement.

At the heart of that revolution lies a simple idea: a decentralised database can be secured by work that is hard to produce and easy to verify. Miners validate transactions, compete to find blocks, and receive newly minted coins and fees as compensation. As the value of those coins increased, so did the mining activity and its associated energy consumption, leading to criticisms on its environmental footprint.

TensorCash starts from a different observation. AI providers are already spending energy and capital on inference that customers value. If that inference can be made efficiently verifiable, the same computation can also serve as proof of work for a decentralised monetary network. The goal is not to force AI products into a blockchain marketplace. It is to let useful inference acquire a second role: securing a scarce digital asset while remaining useful inference.

TABLE OF CONTENTS

I. TensorCash Protocol	3
II. TensorCash Building Blocks	3
III. Related Work	5
IV. Proof-of-Inference and Proof-of-Work	5
i. Introduction	5
ii. Novelty and Challenges	6
iii. Proof of Inference	6
iv. Proof of Work (Consensus)	7
V. Conclusions	8
References	8
Appendix A. Inference Fuzziness and Reproducibility	9
Appendix B. Entropy Collapse and Collision Resistance	9
Appendix C. Attack Vectors Analysis	10
Appendix D. Privacy	11
Appendix E. Inference Example	11

I. TensorCash Protocol

TensorCash is first and foremost a protocol for proof-of-inference. A proof of inference is a compact transcript that lets a verifier check that a declared model was run under declared sampling and precision conditions. This matters beyond mining. An AI provider, inference aggregator, agent platform, or compute-market operator can attach such a transcript to work that would otherwise require a bilateral trusting relationship. The transcript does not make every AI task public and it does not require every model output to become an on-chain object; it simply makes inference portable, contestable, and independently checkable when verification matters.

TensorCash then uses that primitive as proof of work. In the Satoshi design, proof of work turns scarce computation into a consensus signal: valid blocks are hard to produce, easy for the network to check, and chained so that rewriting history requires redoing accumulated work. TensorCash keeps that structure but changes the source of work. Instead of scanning arbitrary nonces until a hash target is met, miners run useful inference on registered open-source models. The inference transcript supplies the unpredictable entropy that ultimately feeds the block-header proof-of-work check.

This paper introduces the overall system: a blockchain with a native token, a proof-of-inference mining process, and a financial ledger that preserves the conservative strengths of a Bitcoin-style architecture while adding native support for AI-secured work, issued assets, privacy-aware compliance, and bounded financial settlement contracts. Companion technical papers specify the verification mathematics, mining-engine integration, core-node modifications, asset protocol, settlement contracts, and coordination layer.

II. TensorCash Building Blocks

The first building block is a verification protocol for autoregressive language-model inference. A miner or AI provider records a bounded transcript of a model run². A verifier can then check that the transcript is consistent with the declared model without requiring bit-for-bit equality across hardware. The verifier is

statistical because production inference is not exactly reproducible across GPUs, kernels, batching regimes, and floating-point formats. The Mining API implements this with a low-overhead capture path inside the sampler, so proof evidence is produced during inference rather than by running a second workload.

The second building block is a Bitcoin Core-derived blockchain. TensorCash preserves the familiar UTXO ledger, peer-to-peer block propagation, permissionless validation, censorship resistance, transaction-fee incentives, and a native coin minted only through proof of work. It also preserves the economic elegance of self-levelling monetary difficulty: if blocks arrive too quickly, the base target becomes harder; if they arrive too slowly, it becomes easier. TensorCash adds an adjusted target, adapted to each model, so different registered models can compete for blocks at the same time while being compensated for their measured inference cost.

The result is a dual-use proof of work. A miner can perform off-chain AI inference on open-source models, serve customers or aggregators, and simultaneously compete for block rewards. For an established AI service, mining becomes an additional revenue stream attached to work it already performs. For a miner without customer traffic, synthetic or aggregator-sourced inference can keep hardware productive while the operator builds demand. In both cases, the network's security budget is tied to computation that can also produce real-world outputs.

Finally, TensorCash adds first-class real-world-asset support at consensus level: issued assets live as UTXO records, signatures commit to asset effects, issuer authority is represented by a rotating on-chain control unit, and regulated transfers can use zero-knowledge KYC proofs so holders prove eligibility without publishing identity. Issuers can also anchor legally operative prospectuses, policy documents, governance records, and holder-only disclosures through on-chain commitments. Native settlement contracts then support spot swaps, repos, forwards, and option-style delivery-versus-payment.

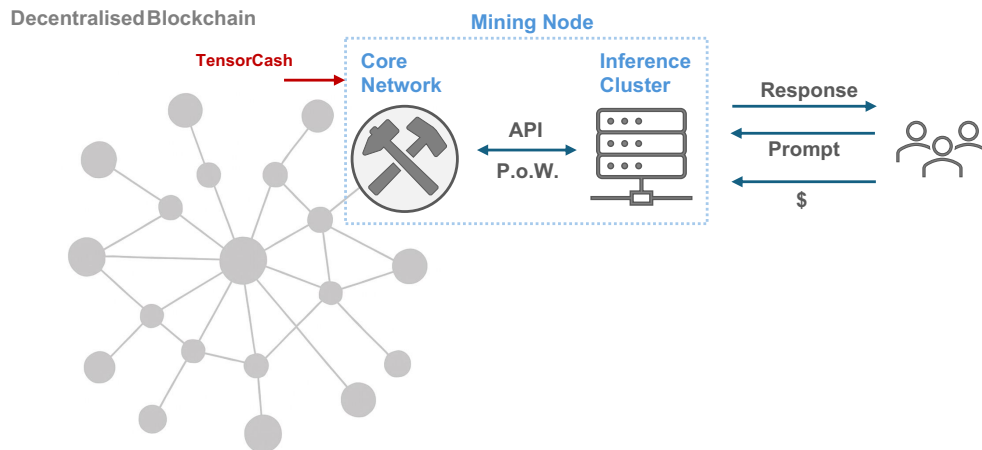


Figure 1. Basic Miner Overview. A Bitcoin-equivalent node interfaces with the miner's inference cluster. The miner serves users for commercial rewards while proof-bearing inference secures the network and earns TensorCash.

2. Model identity, declared precision, prompt and generated token identifiers, deterministic sampling draws, selected logit evidence, statistical summaries, and the chain state used to bind the run.

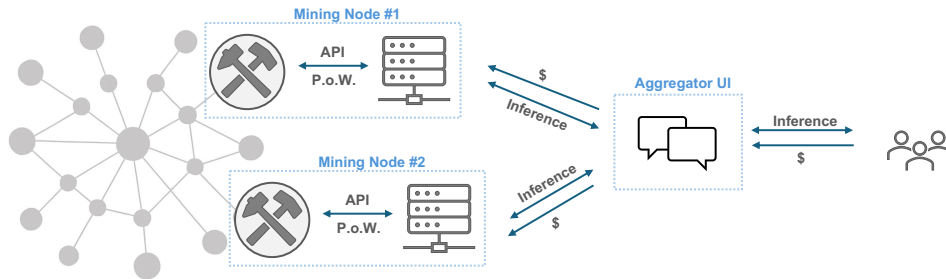


Figure 2. Miners Serving a Commercial Aggregator. Miners without a direct AI customer base can route capacity through an aggregator, keeping the hardware productive while still producing proof-bearing inference.

III. Related Work

Merging AI and cryptography has become an active area of innovation. Several projects aim to bring AI or machine-learning workloads on chain, decentralise compute, or create token incentives around model training and inference. Bittensor uses specialised subnets where participants compete for token rewards under peer evaluation. Gensyn focuses on decentralised compute coordination. EigenLayer-style systems use restaked trust and verification layers for externally executed tasks. These approaches have attracted strong interest despite open questions around scalability, objective task evaluation, and adoption by real production AI providers.³⁴

TensorCash takes a different route. It does not try to own the AI customer relationship or dictate which tasks providers should serve. Market forces already determine which models, latencies, privacy guarantees, and price points users value. TensorCash instead asks whether the inference already being performed can become verifiable and whether that verifiable work can secure a monetary network. The distinction is important for non-crypto-native AI providers: TensorCash should feel like an additional proof and settlement layer around useful inference, not like a requirement to rebuild an AI product inside a blockchain-specific marketplace.

IV. Proof-of-Inference and Proof-of-Work

i. Introduction

Most modern generative AI systems, from chatbots to agents, operate through an autoregressive loop. The user's prompt and context are chopped into tokens. Those tokens are passed through the internals of a model, such as a transformer, and the model outputs a vector of scores for the next token. After normalisa-

3. Zero-knowledge proofs for large-scale models are currently impractical. Generating proofs often adds 10-1000x the baseline compute time and remains infeasible for many large models. Even with techniques like spot-checking, many attack vectors, such as partial model substitution, precision downgrades, or shortcut distillation, remain hard to rule out.

4. While Bittensor has seen growing interest, it faces adoption challenges including limited high-quality subnet diversity, peer-validation gaming, difficulty in objectively measuring model value, and misaligned incentives. Real-world AI deployment also often runs into privacy, cost, and proprietary constraints that open-subnet models do not easily accommodate.

tion, those scores define a probability distribution across the model's token vocabulary. A sampling rule chooses one token from the high-probability region, appends it to the context, and the model repeats the process. This continues until a stop condition is reached.

This probabilistic sampling step is part of what makes the output "generative". The same prompt can produce different valid continuations because the model is not simply returning a fixed database entry; it is repeatedly sampling from a changing probability surface. TensorCash leverages this common paradigm. It replaces ordinary random sampling with pseudorandom sampling generated by a hash function, SHA-256, bound to prior tokens and recent blockchain state. Future sampling entropy cannot be known until earlier sampled tokens exist, and earlier tokens cannot be changed without changing the future path. The concatenation of these entropy outcomes is what gives the proof resistance to collision, reuse, and pre-mining.

ii. Novelty and Challenges

TensorCash sets an intentionally ambitious target for proof of work. Classical proof-of-work systems rely on cryptographic hash functions: they are easy to evaluate, infeasible to reverse, sensitive to tiny input changes, and resistant to collisions. This makes verification simple and the mining game clean.

Inference-based proof of work has a larger attack surface. Miners should be free to choose prompts and contexts that have commercial or research value. They should be able to use admitted open-source models as the field evolves. The protocol should impose minimal performance penalty, should not degrade output quality, and should tolerate ordinary cross-platform numerical variation without creating unfair advantages. These requirements make the system more complex than hash-only mining, but that complexity buys a different property: work that secures the chain can also be useful outside the chain.

The core challenges are determinism and shortcut resistance. Production inference is not bit-for-bit deterministic across hardware, drivers or attention kernels. At the same time, miners must not be able to declare one model and run a cheaper substitute, register hollow models, exploit low-entropy prompts, or precompute completions before the relevant block state exists. TensorCash addresses this by separating proof generation, model admission, per-block verification, and consensus difficulty adjustment.

iii. Proof of Inference

Let f_θ be an admitted autoregressive model with vocabulary V and declared precision dt . At generation step t , the current context is

$$C_t = (x_1, \dots, x_m, y_1, \dots, y_{t-1}),$$

where x are prompt tokens and y are generated tokens. The model emits logits

$$L_t = f_\theta(C_t) \in \mathbb{R}^V.$$

A sampler transforms those logits into a truncated probability distribution

$$\pi_t = S(L_t; \tau, p, k, \rho),$$

where τ is temperature, p is the top- p mass threshold, k is the top- k cap, and ρ is the repetition-penalty parameter. In ordinary inference, the engine draws a random value and selects the token whose cumulative probability interval contains that draw.

TensorCash makes the draw deterministic and transcript-bound. For each step, the miner computes

$$u_t = U(H(\text{TSC} - \text{POI} \parallel B \parallel v \parallel T \parallel t \parallel C_t \parallel dt)),$$

where H is SHA-256, U maps the digest to a uniform value in $[0, 1)$, B is the block-header prefix or equivalent chain-state binding, v is the VDF output, and T is the associated tick count. The chosen token is then

$$y_t = F_{\pi_t}^{-1}(u_t),$$

where $F_{\pi_t}^{-1}$ is the inverse cumulative distribution function of the sampler. The proof object records enough evidence for a verifier to recompute the sampling path and compare the miner's committed statistics against the declared model. The exact transcript schema and statistical gates are specified in the TensorCash Inference Verification whitepaper.

iv. Proof of Work (Consensus)

Proof of inference becomes consensus proof of work when three additional pieces are fixed by the network.

First, the network needs an agreed model registry. A registered model is identified by public metadata such as repository identifier, commit hash, declared precision, content address, and a difficulty scalar. Model admission is not a private operator opinion. A candidate model posts a deposit, enters a registration window, and requires hashpower-visible commit support before it becomes eligible. The audit process estimates effective FLOPs per generated token, checks for hollow or degenerate models, and assigns the model difficulty used by block validation.

Second, the network needs model-aware difficulty. Bitcoin-style base difficulty keeps block cadence stable as total mining capacity rises or falls. TensorCash adds a model-adjusted target so models with different measured inference costs can compete fairly. In simplified form, if T_{base} is the current chain target, D_m is the registered model difficulty, and N is the protocol normaliser, the adjusted target is bounded by

$$T_{\text{adj}} \leq T_{\text{base}} \cdot \frac{N}{D_m}.$$

Smaller or cheaper models therefore face harder hash targets, while larger or more expensive models receive compensating target relief. This adjustment is carried in the block header through `nAdjBits`, allowing multiple admitted models to coexist.

Third, the proof transcript must bind to the block. At fixed token boundaries, the miner hashes the completed proof context. The resulting digest is not a free nonce that can be ground independently; it is valid only if it recomputes from the inference transcript. The block header then uses the derived nonce and the proof commitment in a conventional hash-target check. If the header hash is below the adjusted target, the miner may broadcast the block and its proof. If not, inference continues to the next eligible boundary.

Per-block verification is deliberately tiered. Low-cost deterministic checks reject malformed proofs, unregistered models, invalid sampling draws, bad proof hashes, or headers that do not satisfy the target. Statistical checks then decide whether the transcript is consistent with the declared model and precision. Model-level disputes, such as a later-discovered fraudulent registration, are handled through a slower challenge path rather than by overloading every block validation event.

V. Conclusions

TensorCash joins two systems that have previously lived apart. From AI, it takes the rapidly growing demand for inference and the reality that useful computation is already being performed at industrial scale. From Bitcoin, it takes a conservative consensus architecture: peer-to-peer validation, scarce native issuance, accumulated work, difficulty retargeting, and a ledger whose security does not depend on a central institution.

The protocol's central claim is that AI workloads can be made efficiently verifiable. Once inference is verifiable, it can support decentralised compute markets: providers can prove what they ran, aggregators can contest or audit claims, and users can demand evidence when the cost or trust stakes justify it. Once proof of inference is tied to chain state, that same workload can secure a proof-of-work blockchain without requiring miners to abandon commercially useful AI service.

The result is a self-reinforcing economic loop. Rising TensorCash value can improve AI provider margins and attract more inference capacity into the mining set. More efficient open-source models become economically valuable because their inference can be monetised through both service revenue and block rewards. Compute rewards become more geographically and operationally accessible than specialised hash mining alone. Most importantly, every joule spent on proof of work can also produce real-world inference value.

TensorCash also extends the Bitcoin-style ledger in a practical direction for financial markets. Native assets, zero-knowledge KYC, issuer governance, legally anchored documents, and bounded settlement contracts allow financing and derivative workflows without importing the broad risk surface of a programmable virtual machine. The long-term ambition is therefore not merely an AI coin, but a decentralised settlement network whose security budget is coupled to useful compute and whose financial layer remains simple enough to audit.

References

Nakamoto, S. (2008).

Bitcoin: A Peer-to-Peer Electronic Cash System.

Available at <https://bitcoin.org/bitcoin.pdf>

Appendix A. Inference Fuzziness and Reproducibility

In an ideal world, once a proof is generated, every node would reproduce it bit-for-bit. That is the norm for verifying a hash function. Generative AI inference does not offer this guarantee. A proof here is the product of trillions of floating-point operations, each vulnerable to slight variation based on hardware, kernels, batching, drivers, and inference settings. Bit-for-bit equivalence across platforms is not realistic without sacrificing performance, for example by forcing CPU fallback.

The aim therefore shifts from exact reproducibility to statistical confidence that the computation was performed with the declared model and precision. Lower precision execution is acceptable only when declared and registered as such, with corresponding difficulty adjustment. Undeclared quantisation, surrogate models, and shortcut implementations should create structured shifts that differ from ordinary platform noise.

The verifier addresses this through stepwise replay and statistical comparison. Instead of accepting avalanche divergence across a long generation, it checks individual model steps against the previously sampled context and compares committed logit summaries against recomputed values. Cross-platform deviations are treated as noise; quantisation and substitution introduce systematic shifts and quality loss. The full treatment belongs in the technical v1 protocol paper, especially the "Full Verification: Replay under Platform Noise" subsection of

TensorCash Inference Verification: Mathematical Specification

Appendix B. Entropy Collapse and Collision Resistance

Generative models often run into the billions of parameters, but their computational complexity does not automatically translate into output diversity. A capable model may produce highly confident completions for famous quotations, boilerplate, or tasks with a nearly deterministic answer. Users may also select decoding settings that strongly favour the highest-probability token at each step.

This determinism creates a proof-of-work problem. If output paths collapse to near certainty, they become easier to reuse, predict, or search around. TensorCash therefore enforces an average sampled-token probability ceiling for accepted proof windows. The exact value and rationale are specified in the TensorCash Inference Verification whitepaper; the economic intuition is simple: work that leaves no meaningful sampling entropy should not receive the same mining value as inference that creates a fresh, transcript-bound path.

Assuming this mild entropy constraint, collision resistance follows from the recursive nature of generation. Each new token updates the context, each context changes future sampling draws, and the chance of two long valid transcripts matching token-for-token decays rapidly as sequence length grows.

Figure B1. Collision Probability

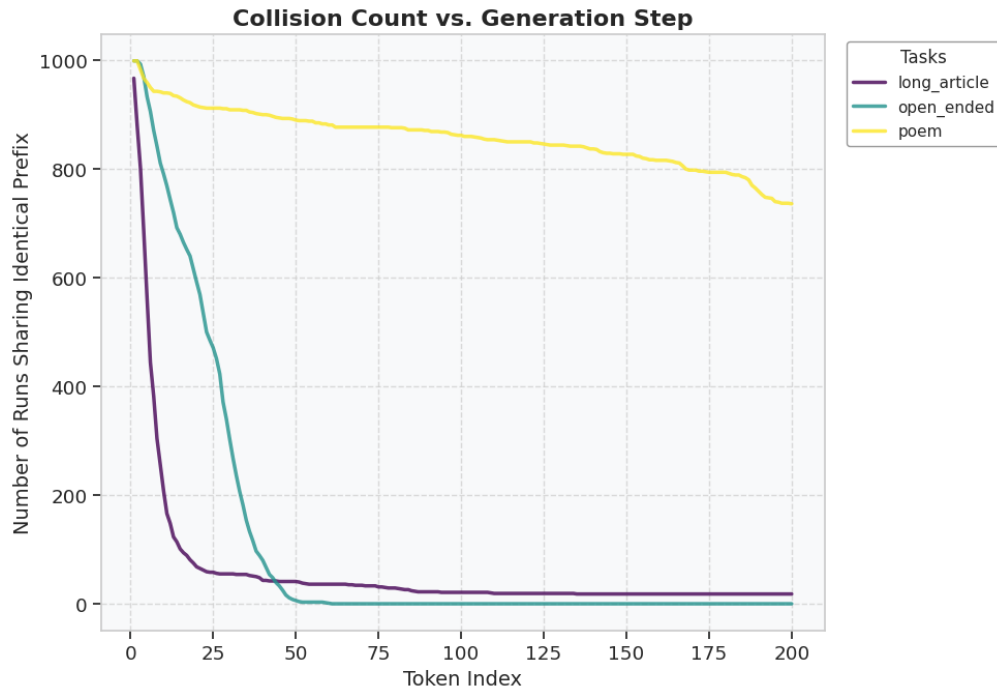
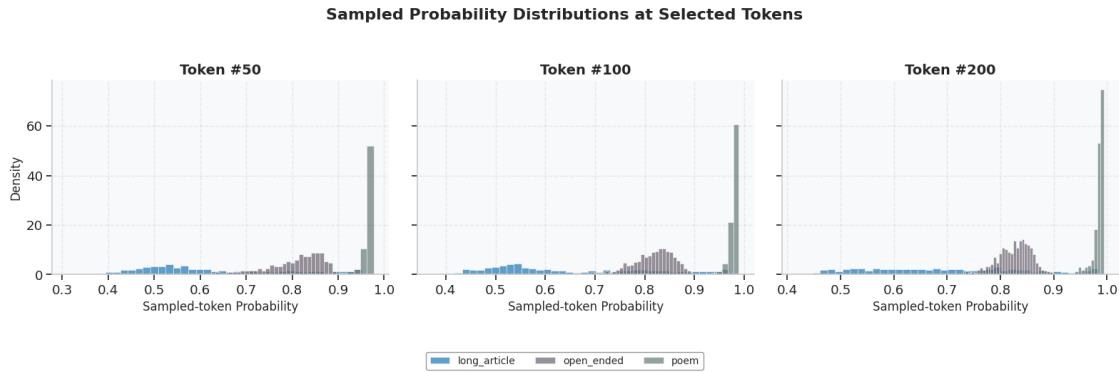


Figure B2. Entropy Requirements



Appendix C. Attack Vectors Analysis

Designing a proof-of-work scheme grounded in generative AI introduces a wider attack surface than classical hash mining. The main attack classes are summarised here; implementation-specific verifier details are deferred to the TensorCash Inference Verification whitepaper.

Attack vector	Threat	Protocol response
Precomputation and prompt banks	Miners may precompute prompts or completions before the next block and hope one later satisfies the target.	Sampling binds to current chain state: header prefix, VDF output, tick, rolling context, precision tag, and final transcript. A banked prompt may still be used, but its proof value is unknowable without re-execution.
Shortcut search	An attacker may branch through contexts, create loops, or stop early when a partial transcript appears promising.	The proof window must complete before the boundary hash is meaningful. The block nonce derives from the completed transcript, so partial paths and independent nonce grinding do not validate.
Low-entropy degeneracy		Deterministic sampling entropy is injected at each step, and accepted windows must satisfy an average sampled-probability ceiling.

Attack vector	Threat	Protocol response
	Low-temperature or deterministic completions can reduce proof search to predictable text replay.	Useful low-entropy inference can exist, but it should not become an easy mining shortcut.
Model fraud	A miner may register a model that advertises high cost while routing through cheap inactive paths.	Model admission uses measured FLOPs per token, sparsity- and dtype-aware accounting, validity probes, file-size sanity checks, deposits, and hashpower-visible commits during registration.
Execution mismatch	A miner may use a cheaper model, lower precision, or a surrogate while declaring a more expensive model.	Per-block verification compares committed transcript statistics against recomputed model behaviour under a platform-noise envelope. Honest variation should look like noise; substitution or undeclared quantisation should create structured drift.
Time and withholding attacks	Sudden compute bursts, cheap header-only forks, and selfish mining attack the chain rather than a single transcript.	A VDF tick structure anchors block progression to sequential time, and proof transcripts bind to declared VDF state. Withholding remains possible, but pre-mining future valid proofs is materially harder than hiding hash work.

Appendix D. Privacy

TensorCash requires open-source weights and reproducible proof transcripts, so privacy must be framed carefully. There are three relevant privacy vectors: model weights and architecture, training data, and inference data. The first is public by design. The second is a broader model-safety problem. The third is the direct issue for proof of inference, because prompts and context can contain personal or commercially sensitive information.

The proof mechanism does not require plain-text prompt disclosure as such; it works over token IDs, token contexts, and logit summaries. However, if the tokenizer, vocabulary, or codec is public, token sequences can often be converted back into intelligible text. This is acceptable for public document summarisation, public search, public agents, and many commercial workloads, but unacceptable for sensitive personal use cases.

The practical mitigating factor is miner control. Only a small fraction of inference segments become successful block proofs, and the miner controls which prompts are eligible for submission. Sensitive customer prompts can be excluded from mining proofs, while non-sensitive, public, synthetic, or consented workloads can be routed into proof-bearing generation. More advanced privacy-preserving inference methods, such as private tokenizers, obfuscated vocabularies, non-linear encoders, secure hardware, or cryptographic private inference, are future-work directions rather than assumptions of the reference protocol.

Appendix E. Inference Example

TensorCash aims to make the cryptographic layer invisible to end users and minimally invasive for inference providers. A successful proof-bearing completion should look like ordinary model output, and the provider's workload should remain close to standard generation.

Prompt

"Explain why a dual proof of work scheme, where the actual work has real world utility, is potentially valuable."

Example Output

Model: [mistralai/Mistral-7B-Instruct-v0.3@e0bc86c23ce5aae1db576c8cca6f06f1f73af2db](https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3@e0bc86c23ce5aae1db576c8cca6f06f1f73af2db)

A dual proof-of-work scheme with real-world utility can be valuable because it links network security to useful production. Miners still compete to create scarce, hard-to-forge work, but the work also serves users: answering queries, producing summaries, generating media, or supporting other inference tasks. This improves the economic story of proof of work, because the energy spent securing the chain also produces an output people already demand. It can also attract AI providers into the mining set, strengthening security while giving those providers an additional revenue stream.

The important property is that the cryptographic transcript does not require a separate artificial computation. It rides on the inference path itself. Throughput measurements on early implementations showed roughly unchanged single-batch generation speed on an L40S-class GPU, with the remaining overhead dominated by proof collection and verification plumbing rather than by the sampling modification itself.