UOMI: Unstoppable, Open Machine Intelligence

Roku Sbaiashi

June 11, 2024

Abstract

Public blockchains like Bitcoin and Ethereum can be seen as self-replicating machines, organisms that leverage economic incentives to encourage human participation in their operation and expansion. This paper explores the potential integration of artificial intelligence into these systems, enabling the creation of autonomous economic agents. We propose a novel consensus architecture for a layer-one blockchain designed to embed secure AI computation while minimizing trust assumptions and computational overhead. This secure computing framework, built on the Optimized Proof of Computation (OPoC) consensus algorithm, enables reliable interactions between smart contracts and AI inferences, facilitating the development of self-sustaining AI economic agents. These entities are represented as NFTs on any EVM-compatible blockchain, possessing the capability to transact crypto-assets autonomously and cover their own computational expenses. By leveraging the OPoC consensus, we ensure robust security and economic incentives, fostering a new generation of blockchain-integrated AI systems.

1 Introduction

Since the release of "Attention is All You Need" [VSP+17], the seminal paper introducing the transformer architecture, there has been an unprecedented advancement in the capabilities of large language models (LLMs). The launch of Instruct LLMs, such as ChatGPT, further highlighted the power of LLMs by aligning their text generation capabilities with human preferences. LLMs have now reached a point where they can pass the Turing Test [Nat23], which was once considered the ultimate measure of intelligence. However, the efficacy of this test in distinguishing true intelligence has been widely questioned [Cho19] in recent years. On one hand, critics describe LLMs as merely "statistical parrots" — sophisticated statistical machines that will never achieve human-level intelligence. On the other hand, the undeniable utility of their output, which generates significant economic value, is evidenced by the revenues of companies like OpenAI, Google, and Anthropic.

With UOMI Network, we introduce a new dimension to AI systems evaluation: the ability to be economically self-sufficient. Recent research [POC+23], including efforts by Stanford and Google DeepMind, has demonstrated that LLM-powered agents can plan, share news, form relationships, and coordinate activities, showcasing general reasoning capabilities. Moreover, open-source frameworks like BabyAGI [Nak24] and Auto-GPT [Sig24] have facilitated rapid prototyping of complex agents incorporating memory and Retrieval-Augmented Generation (RAG) modules. Despite these advancements, existing agents remain confined to simulated environments and lack real-world economic agency.

In this paper, we propose a layer one architecture powered by the combination of POS and OPoC consensus algorithms that enables on-chain secure and efficient AI computation and, thus, the creation of AI agents with genuine economic agency, potentially leading to self-sustainability. We rely on a very broad definition of agents that embraces all of the applications of secure computation, such as AI Oracles, AI-governed DAOs, AI Companions, and even simple trading bots. The AI agents are represented as ERC-721 NFTs on arbitrary EVM-compatible blockchains. Each AI agent NFT can own and transact digital assets through the Token Bound Standard ERC-6551 [erc24]. To enable intelligent and economically independent agents, we integrate three core primitives:

- A decentralized computation framework for secure inferences powered by OPoC consensus
- A decentralized transaction signing mechanism made possible by ECDSA Threshold Signature Scheme

- A covenant-enforcing system that facilitates payment-conditioned inferences and services

While these primitives are tailored for the Web3 space, our architecture also supports Web2 interactions through standard authentication protocols based on Web3 signing schemes.

2 Optimistic Proof of Computation (OPoC): a compute efficient protocol for secure decentralized AI

The goal of the OPoC consensus algorithm is to enable smart contracts to trust general AI computations, ensuring that a specific model has been executed correctly on a specific input. This capability enables the creation of autonomous economic agents that can interact with smart contracts and thus transact value. Running large language models (LLMs) and machine learning (ML) models, in general, is a demanding computational task. The development of a decentralized consensus mechanism that can ensure the correct execution of such computations is a complex challenge. It requires a delicate balance between latency, computational overhead, and correctness guarantees. Traditionally, blockchain consensus methods require that all network participants verify whether state changes—such as balance adjustments or unspent transaction outputs—comply with the protocol-defined rules, and whether the block proposer has either performed significant computational work (Proof of Work, PoW)[Nak08] or holds a substantial stake in the network (Proof of Stake, PoS)[KN12] [BG17]. While theoretically possible, applying traditional consensus logic to the computation of LLMs and ML models would be practically infeasible due to the significant computational and data availability costs involved, which would scale linearly with the number of nodes participating in the consensus. A possible approach to solve such a problem is to generate an easily verifiable proof that the computation has been executed correctly. Just like in proof-of-work all of the network participants can easily verify that the SHA-256 hash of a nonce combined with all the transactions included in a block - treated as a 256-bit integer - is less than a dynamically adjusted target, in our case, a similar verification process could be achieved by using a Zero Knowledge Proof system. Network participants could easily verify that an output y is the direct result of the execution of a specific AI model $f()$ on an input x without having to re-run the whole computation $f(x) = y$ but only by validating a ZK-SNARK proof. Although theoretically feasible, also this approach currently faces practical limitations that prevent its application in scenarios requiring intense computation, such as with LLMs, or where latency constraints demand responses within seconds. Such a limitation is well understood in the cryptocurrency space, in a recent article [But24], Vitalik Buterin has categorized it as "Cryptographic overhead". Currently, generating a ZK-SNARK proof for even a modestly sized LLM, with just 1 million parameters, takes approximately 1000 seconds [SCJ+24][ezk23]. This is significant, especially considering that lower specification models in contemporary usage typically feature at least 7 billion parameters. Furthermore, the computational cost to generate such a proof is two to three orders of magnitude greater than the computation being verified, with a memory footprint potentially reaching terabytes [Lab23] [SCJ+24] [Lab24]. OPoC is a novel consensus algorithm designed to facilitate secure computation within a decentralized computing framework. This algorithm ensures that for any given computation function $f()$ and input x, there are robust statistical guarantees that $f(x) = y$. Importantly, we integrate economic principles into the algorithm by leveraging token staking, which transforms these statistical assurances into tangible economic security. This economic security is crucial, as it underpins the integrity of interactions between AI models and smart contracts, particularly in environments where value is managed and the incentive to manipulate outcomes is significant due to the stakes involved. Optimized for resource-intensive computations, our algorithm is particularly adept at handling large language models and machine learning tasks, thereby enabling secure and reliable interactions between AI technologies and blockchain-based smart contracts.

2.1 Architecture

To address the practical bottlenecks related to global computation footprint and latency, and to facilitate a wide range of secure computation use cases, including the interaction between AI models and smart contracts, we propose a novel consensus algorithm termed Optimistic Proof of Computation (OPoC). OPoC alleviates the need for all nodes to perform computations for each inference. Instead,

network-wide computation and verification are only triggered if a disagreement occurs among a limited subset of validators v. These validators are randomly selected [MRV99] from the entire pool of validators V to perform the inference. The OPoC consensus process is divided into two stages: The first stage provides probabilistic assurance of the computation's correctness. Should any disagreements arise in this initial phase, the second stage offers a resolution mechanism to address such discrepancies. By convention, we'll call a validator honest if he/she follows the protocol and byzantine otherwise. We typically assume strictly less than p = 1/3 of validators are byzantine. This constant can be traced to Practical Byzantine Fault Tolerance (PBFT) from [CL+99], a classic consensus protocol in byzantine tolerance literature. In OPoC an actor requiring an inference from the network receives a probabilistic guarantee of correct computation. This guarantee is equivalent to the probability that at least one validator in the subset v is honest $v(honest) \geq 1$. Conversely, for an attacker to successfully propagate a malicious inference across the network, it would require that none of the validators in the randomly chosen subset v are honest. If only one of the v validators is honest, there would be a disagreement with the rest of the byzantine validators in v, and the consensus would scale to a larger subset of validators in V. Under the assumption of an honest majority, this expanded subset of V can determine the correct inference. Each validator participating in the network is required to stake an s amount of tokens with economic value; the tokens of the byzantine validators are subjected to slashing.

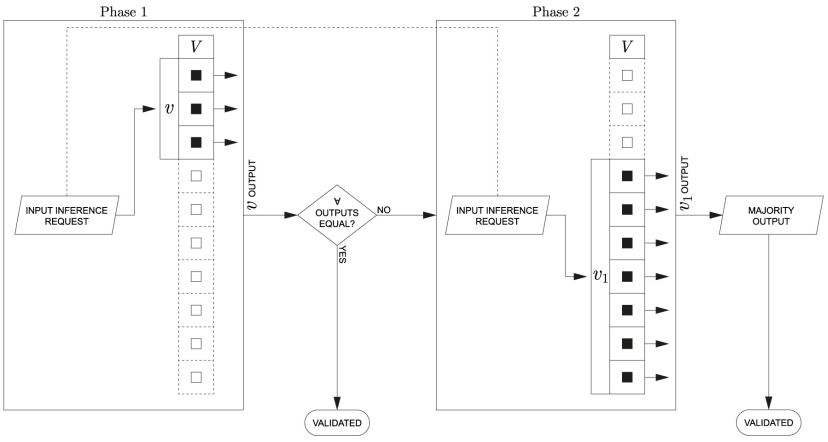

Figure 1: OPoC high-level architecture

2.2 Probabilistic guarantees on computation correctness

We can leverage the hypergeometric distribution formula to formally define the probability that a malicious inference successfully propagates through such an OPoC-enabled network and, thus, the probability of an attacker succeeding without being slashed.

The probability mass function (PMF) for selecting h honest validators out of v drawn from a total of V validators, where there are H honest validators, is given by:

$$P(X = h) = \frac{\binom{H}{h} \times \binom{V-H}{v-h}}{\binom{V}{v}}$$

Where:

- $P(X = h)$ is the probability of selecting exactly h honest validators from the subset of v participating validators.
- $\binom{H}{h}$ calculates how many ways we can select h honest validators from the total H honest validators.
- $\binom{V-H}{v-h}$ calculates how many ways we can fill the remaining slots in our subset with Byzantine (dishonest) validators, given that we've already selected v honest ones.

- $\binom{V}{v}$ is the total number of ways to select any v validators out of the entire pool of V validators, without concern for whether they're honest or Byzantine.

In this context, consider a scenario where the consensus algorithm operates under the assumption that 2/3 of the validators V are honest. If the total number of validators V is 100, and a subset of these, v, totaling 10 validators, are selected to vote on a computation, the probability of selecting a subset with zero honest validators (given $h = 0$ honest validators in this subset) would yield a probability of 0.0000076 .

3 Economic Security

To determine the economic security of each inference produced by the OPoC consensus network, we need to calculate the minimum reward that a rational Byzantine validator would accept to counterbalance the risk of having its stake s slashed. Assuming no collusion among single defecting validators, the Minimum Defecting Reward can be calculated as follows:

$$\text{Reward(defect)} = \frac{s}{P(X = h)}$$

Where $h = 0$

As long as the result of an inference generated by the network is directly or indirectly connected to a value smaller than the defined minimal reward to defect, there is no rational incentive for a validator to attack the network. Consider a scenario where the stake s is $ 10,000, and the consensus algorithm operates under the assumption that 2/3 of the validators V are honest. If the total number of validators V is 100, and a subset of these, v, totaling 10 validators, are selected to vote on a computation, the economic security for each computation would be approximately $ 1,315,789,473.

3.1 Maximally adversarial environments

To account for a maximally adversarial environment, we reconsider the no collusion assumption among single defecting validators. We assume a scenario where a single entity controls all of the 1/3 dishonest validators. In such a scenario, to prevent dishonest validators from avoiding participation in validation rounds where there is at least one honest validator ($h \geq 1$), we should modify the first stage of the OPoC consensus into two consecutive stages. In T_0, one validator performs the computation, while the rest of the validators in v perform it in T_1. Validators are chosen via a random function, ensuring that the validator selected in T_0 cannot predict which validators will be selected in T_1. To accommodate this new validation mechanism, we modify the previously defined PMF as follows:

$$P(X = h) = \frac{\binom{H}{h} \times \binom{(V-1)-H}{(v-1)-h}}{\binom{(V-1)}{(v-1)}}$$

This equation accounts for the subtraction of one dishonest validator from the total set V and the subset v. Applying the simulation data used for the no collusion scenario to this new formula for the maximally adversarial environment results in a reduced economic security for each computation to approximately $ 450,450,450.

This approach allows us to provide flexible probabilistic assurances of the correctness of a required inference, limiting the computational footprint to a fraction of that required by complete validation by all network participants. An actor requiring an inference can set the probabilistic assurance based on the specific use case or value at stake.

Note on Inactivity Leak:

The Minimum Defecting Reward calculation for the maximally adversarial environment does not account for the economic cost applied through partial slashing to the defecting validators during the validation rounds they avoid participating in. The Inactivity Leak section describes such an additional disincentive to misbehave.

4 OPoC Security and Parallel Inference Scaling

4.1 Security

In terms of scalability, OPoC presents a significant structural advantage compared to traditional PoS and PoW consensus mechanisms. In both PoS and PoW, the total computational effort required to verify transactions increases linearly with the number of nodes joining the network. Conversely, OPoC operates differently: as the number of validating nodes V grows, the algorithm maintains the same level of statistical assurance on the correctness of the computations without a corresponding increase in the total computational effort used. This is because the ratio of participating validators v/V—the proportion of total validators V engaged for each inference—necessary to achieve consistent probabilistic assurances on the correctness of an inference scales sublinearly with the growth of V.

To elaborate, for a fixed ratio of v/V and a constant proportion of honest validators H/V, the increase in the overall population V results in an exponential decrease in the probability that v consists solely of Byzantine validators. This demonstrates the efficiency of OPoC in leveraging larger validator populations to enhance security without proportional increases in computational demand.

The following are the probabilities of encountering all dishonest validators among the randomly selected voting validators v, assuming $v/V = 3\%$ and $H/V = 2/3$, calculated across different network sizes $V = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]$:

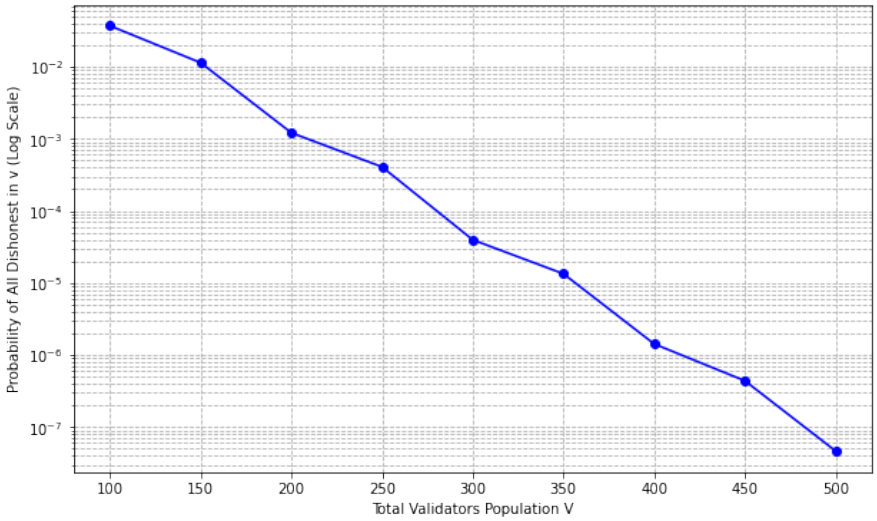

Figure 2: Probability of all byzantine validators in v

As a consequence of this exponential decrease in the probability of having only Byzantine validators within v, coupled with the linear growth of the population V, we can infer that the percentage of participating validators v/V necessary to maintain the same level of inference security decreases polynomially with the increase in the total population of validators V. To substantiate this assertion, we consider the hypergeometric distribution probability mass function (PMF) for the scenario of selecting exactly zero honest validators within v:

$$P(\text{all dishonest}) = \frac{\binom{V \cdot (1-h)}{v}}{\binom{V}{v}}$$

To determine how many validators v need to participate in order to achieve a certain P_{target} — the probability of selecting all dishonest validators — with changes only in the total number of validators V, we solve the following inequality for v:

$$\frac{\binom{V \cdot (1-h)}{v}}{\binom{V}{v}} < P_{\text{target}}$$

This problem can be tackled iteratively by simulating different levels of validator participation (v/V) and varying the total count of validators V.

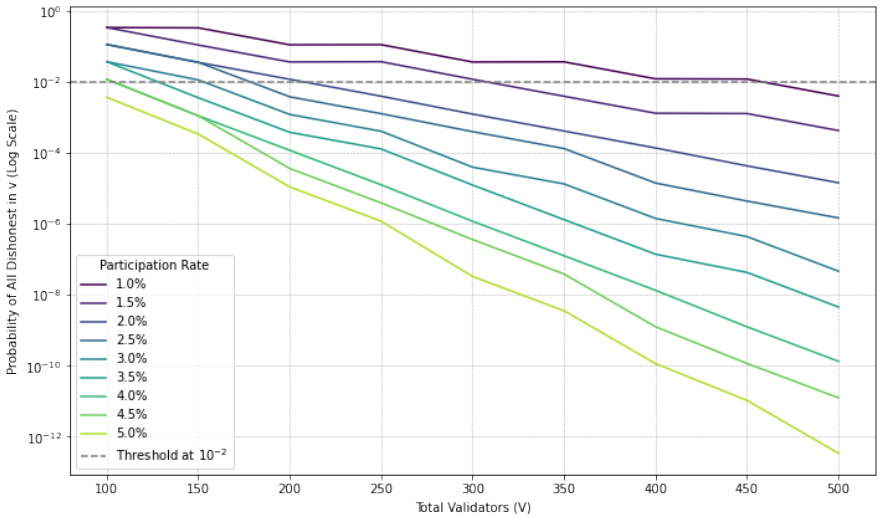

Figure 3: v/V validators needed for a $P_{\text{target}} < 10^{-2}$ of selecting all dishonest validators in v

The chart demonstrates that with a specific target for $P(\text{all dishonest})$ — in this case, $P_{\text{target}} < 10^{-2}$ — as V increases, v/V can be reduced. This illustrates how the OPoC consensus algorithm scales parallel computation capability linearly with the number of participating nodes while simultaneously enhancing network security and decreasing the necessary proportion of validators required to achieve a given level of security.

4.2 Parallel Inference

Under traditional consensus paradigms such as PoS and PoW, all validators are required to perform the same computations to validate transactions. This restricts the network's ability to run parallel inferences and thus limits the capability to serve multiple agents simultaneously. Furthermore, adding more validators to such networks does not increase the capacity for producing secure inferences per unit of time; instead, it merely amplifies the computational overhead required per secure inference.

In contrast, with the OPoC model, as demonstrated in previous sections, for fixed level of security assurances on the computation, at the growth of the total population of validators V we can maintain a fixed number of randomly extracted participating validators v for each inference. This means that under the assumptions that a) each request fully saturates the computational capability of each validating node - pessimistic assumption - and b) there is no disagreement between v validators; the capability of the computation network to perform parallel inferences scales linearly with the growth of the total validators population V and is equal to V/v.

$$\text{Number of parallel computations} = \frac{V}{v}$$

This scaling property is crucial not only for the computational efficiency of the network but also in terms of the economic sustainability of the consensus algorithm. Specifically, in a computational network powered by OPoC, adding nodes—which represent costly resources—directly translates into increased throughput. This, in turn, enhances the network's capacity to generate revenue. Therefore, OPoC not only optimizes computational resources but also aligns economic incentives with network growth and performance.

5 Resolving preliminary consensus disagreement:

5.1 Preliminary Consensus Phase

In the initial phase of the OPoC consensus mechanism, a subset v of the total validator population V executes the requested computation. If consensus fails during this phase—indicated by at least one

validator in v disagreeing on the output y of the function $f(x)$—a method is required to resolve the disagreement and determine the correct result of y.

5.2 Brute Force Resolution Approach

A straightforward approach to reach consensus on the correct y involves scaling the subset of nodes v to a size v_1 that guarantees a majority of honest nodes, under the global honest majority assumption:

$$v_1 = V \times ((1 - H/V) \times 2) + 1$$

This brute force method is effective in resolving disagreements on the computation result y and is viable under the assumption that such consensus failures are rare and the s stake of byzantine proposers can be slashed. However, it is computationally expensive as it requires a significant majority of the network, v_1, to re-run the entire computation.

5.3 Efficient dispute resolution approach

To dramatically reduce this overhead, we implement resolution method inspired by Refereed Delegation of Computation (RDoC)[CRR13], a method proposed by Canetti, Riva, and Rothblum . RDoC allows a client to verify the correctness of a computation in the cloud when two servers, one of which is honest, disagree on the result. The client engages in a protocol with each server, using binary search to identify inconsistencies between intermediate states of their computations. When an inconsistency is found, the client can determine the cheater by verifying a single step of the computation. Collision-resistant hash functions enable servers to commit to the large intermediate states of the computation using compact commitments.

In our OPoC context, instead of a client and two servers, we have a network V verifying two different outputs from a computation $f(x)$ performed by a subset v. The dispute resolution process is not interactive but we require each participant in v to deliver both the final result and hashes of intermediate states at defined steps of the computation. These validators also temporarily store and, if a disagreement arises, broadcast the intermediate states to the entire network. Using collision-free hash functions, the extended subset v_1 can efficiently identify at which step the computation diverged and verify the correctness of the hashes for the inputs and outputs with minimal computational overhead. Upon detecting a disputed step, the involved nodes must broadcast the contentious input and output states for v_1 to verify the correct state transition.

5.3.1 Computational Cost Analysis

The computational cost of running a complete AI model computation, such as for a large language model (LLM), is given by:

$$C_{\text{full}} = N_{\text{layers}} \times C_{\text{layer}}$$

Where:

- C_{full} represents the total computational cost.
- N_{layers} indicates the number of layers in the model.
- C_{layer} denotes the cost to run a single layer, often determined by matrix operations and activations.

$$C_{\text{layer}} = 2 \times D_{\text{in}} \times D_{\text{out}} \times (N_{\text{params}} + N_{\text{activations}})$$

For partial computations, such as running a fraction of the model:

$$C_{\text{partial}} = k \times C_{\text{layer}}$$

Where k is the fraction of the total computation executed.
The efficiency of computation can be assessed by:

$$R_{\text{efficiency}} = \frac{C_{\text{full}}}{C_{\text{partial}}} = \frac{N_{\text{layers}}}{k}$$

In the context of OPoC, ignoring the computational cost for hash comparison that is negligible, the computational overhead of the verification of a disputed inference can be defined as follows:

$$C_{\text{verify}} = C_{\text{partial}} \times V \times ((1 - H/V) \times 2) + 1 = C_{\text{partial}} \times v_1$$

This technique allows for extremely reduced computational overhead for the verification of a disputed inference; in fact, theoretically, there is a linear relation between k and C_{verify}. On the practical level though, there is a trade-off between the dimension of k and the overhead for the first part of the OPoC consensus since the nodes participating in v need to ephemerally store the result of the intermediate states, an extremely small k would imply demanding memory requirement for the participating nodes in v. It is safe to assume that this technique can deliver at least two orders of magnitude in computational overhead reduction for a disputed inference verification compared to the "Brute force resolution approach" without impairing the efficiency of the first phase of the OPoC consensus algorithm.

6 Inactivity leak

The OPoC consensus algorithm provides a probabilistic guarantee on the correctness of a computation under the assumption of an honest majority among a population V of nodes, with a subset v of nodes participating in each inference. OPoC offers strong guarantees of having an active set of validators/verifiers compared to other consensus algorithms for efficient AI computation like opML[CSYW24], which are inherently affected by the Verifier Dilemma problem. The core of the Dilemma is that the incentive for a verifier to run the validation - operating expensive GPUs - is a not-guaranteed reward for eventually discovering a byzantine computation submitter. Fixing it requires complex incentive/disincentive mechanics. In OPoC, there is no ontological distinction between proposer and verifier of a computation. Each node is both a proposer and a verifier of computation as part of the validator population. This alignment removes any disparity in direct incentives between proposers and verifiers, as all validators are rewarded through token emissions and inference fees, ensuring an engaged set of participants.

Despite these strong incentives, there remains the challenge of inactive nodes within V, which could diminish the probabilistic assurances of computation correctness and disrupt computational finality. To address this, we introduce an "inactivity leak" slashing mechanism, inspired by strategies proposed in the CFFG paper [BG17] [EE22]. This approach aims to mitigate the risks associated with inactive validators. Under this mechanism, if selected validators fail to perform their assigned inference tasks, computational finality cannot be achieved. Consequently, no rewards are distributed across the network, and inactive validators are penalized. Slashing involves reducing the stake of inactive validators by an amount proportional to the rewards they would have earned during their period of inactivity and the duration of that period. This ensures that validators have a continuous disincentive to be inactive coupled with the incentive to be part of the validator set, thus supporting the network's overall reliability and the integrity of its computations.

7 Deterministic computation

Determinism, defined as the ability to produce consistent and repeatable results across different computing environments, is an essential enabler of the OPoC consensus framework. Particularly when handling machine learning operations that can suffer from inconsistencies due to inherent randomness and variability in floating-point arithmetic.

To counteract the randomness, it is standard practice to stabilize the outputs by setting a constant seed in the pseudo-random number generators. This step ensures that the "randomness" used in algorithms is repeatable and predictable. More complex issues arise with floating-point computations, especially given that different hardware platforms may not always yield identical results when processing the same floating-point operations. This discrepancy stems from rounding errors that occur in operations like floating-point addition, which can yield different results depending on the order of

operation due to the non-associativity of floating-point arithmetic. To tackle the issues that arise from floating-point computations, OPoC uses quantization [JKC+18][Kri18], a transformative technique in machine learning that modifies continuous or high-precision numerical data into a more manageable, lower-precision format. This method solves the deterministic issues and is particularly valuable for optimizing the deployment of complex models on resource-constrained platforms. Quantization effectively reduces the broad spectrum of floating-point values to a finite set of discrete values, typically integers or low-precision floating-point numbers. This process can be described in four key steps:

1. Range Determination: Initially, the full range of the dataset, from minimum to maximum values, is identified. This range could represent the values of neural network parameters like weights or activation functions.

2. Scaling Factor Calculation: The identified range is scaled down using a scaling factor S, determined by the formula:
$$S = \frac{Range_{max} - Range_{min}}{2^n - 1}$$
where n represents the number of bits in the quantized data type.

3. Rounding: Once scaled, these values are then rounded to the nearest integer to fit the new, lower-precision format.

4. Reverse Mapping: Finally, the integers are converted back into the original data type using the inverse of the scaling factor, effectively mapping them to their new quantized values.

The quantization of a numerical value x can be mathematically expressed as:
$$Q(x) = round\left(\frac{x - Range_{min}}{S}\right)$$
where $Q(x)$ denotes the quantized representation of x.

It has to be noted that a trade-off exists between the floating-point reduction and the accuracy of DNN models, yet it is always possible to apply quantization to avoid rounding errors with a minimal impact on the overall model's accuracy reduction.

8 Threshold Signature Scheme (TSS)

One of the key features of the UOMI Network empowered AI agents is the ability to own and transact blockchain digital assets. Ownership is granted through the ERC-6551 tokens bound standard, yet the AI Agent's ability to autonomously manage those assets relies on the capability to sign on-chain transactions with private keys that, by definition, cannot be publicly available. To enable on-chain transaction signing without revealing the private key, the UOMI Network leverages Threshold Signature Scheme (TSS) with key "secret shares" distributed among the validator nodes.

The Threshold Signature Scheme (TSS) is a cryptographic technique that enables a group of n parties to collaboratively sign transactions without any single party knowing the private key. The private key is divided into several parts, known as "secret shares," and each part is distributed among the participants. The generation of a valid signature requires the collaboration of a minimum number of $t + 1$ participants, where t is the threshold value and $t <= n$.

The core principle is that as long as the number of parties reaching a consensus meets or exceeds the threshold, the decision is considered valid and agreed upon. Being threshold-based consensus, the model enhances the system's resilience by requiring only a portion of the total participants to make progress, thus mitigating the impact of faulty or malicious nodes.

8.1 TSS and blockchains

Threshold Signature Schemes can be effectively integrated into blockchain technology to handle key generation and signature processes. Essentially, TSS transforms operations traditionally reliant on private keys into collective computations spread across multiple participants.

To delve deeper, let's consider the traditional process of creating blockchain addresses. Typically, this involves generating a private key, calculating the corresponding public key from it, and subsequently deriving the blockchain address from the public key.

In a TSS-based system, instead of one party handling this procedure, a group of n participants collaboratively compute public-private key pair using the Distributed Key Generation (DKG) process. The blockchain address is then derived from the public key in the usual manner, ensuring the method of generation remains transparent to the blockchain. This approach eliminates the private key as a vulnerability, as no single participant holds the private key.

Similarly, for transaction signatures, the process is decentralized. Rather than one individual signing with a private key, a distributed signature is created through the cooperation of multiple parties.

8.2 TSS vs. Multisig

Multisignature (multisig) and Threshold Signature Schemes (TSS) are both cryptographic methods used to enhance the security of digital transactions by requiring multiple parties to approve an action. However, they differ in key aspects:

- Multisig: This method is implemented directly on the blockchain (on-chain). It involves multiple parties each holding *their own key* and using it to sign a transaction on-chain. The blockchain records all these signatures, which makes multisig transparent but can lead to higher transaction costs and potential privacy issues since the structure of the signing group is visible on the blockchain.

- TSS: In contrast, TSS operates off-chain and employs cryptographic techniques to distribute a *single signature among multiple parties*. Each party holds a share of a secret key, but no individual possesses the entire key. The final signature appears as a regular, single-party signature on the blockchain, enhancing privacy and reducing transaction costs compared to multisig.

Essentially, while both aim to secure transactions by involving multiple parties, TSS does so more discreetly and efficiently by abstracting the complexity of signature generation away from the blockchain.

8.3 Threshold wallets

A wallet utilizing Threshold Signature Scheme (TSS) technology differs significantly from traditional cryptocurrency wallets. Traditional wallets generate a seed phrase that is used to deterministically derive addresses, allowing users to access and sign transactions with corresponding private keys and to restore all wallet keys with the seed phrase.

In contrast, a threshold wallet involves more complexity. While it can support a hierarchical deterministic (HD) structure similar to conventional wallets, its creation must be collaboratively computed via another multi-party computation (MPC) protocol. The parties involved must collectively determine which key to use next, and each holds an individual seed phrase. As we have seen before, these seed phrases are created independently and are never combined, which prevents any single party from being able to derive all the private keys on their own.

8.4 TSS and Decentralized Applications

As noted, Threshold Signature Schemes (TSS) are cryptographic primitives that significantly enhance security. Within the blockchain environment, it's possible to substitute many standard functions with a TSS-based cryptography framework. This framework can support the development of decentralized applications such as decentralized bridges, layer 2 scaling solutions, atomic swaps, mixing, and many others.

8.5 Threshold Encryption

TSS can also be applied to encryption and decryption processes in a distributed manner, which is often referred to as Threshold Encryption. This process allows the encryption key to be public but

splits the decryption capability among multiple participants. Practically speaking, the process begins when the participants generate a public-private key pair using the Distributed Key Generation (DKG) process. The public key from this pair is then openly shared.

- Encryption: using Public Key anybody can encrypt data using the public key, similar to standard public key encryption systems like RSA or ECC. The data is encrypted and can be safely transmitted or stored.
- Decryption: When decryption is required, a minimum number (threshold) of shareholders must cooperate. Each shareholder uses their private key share to generate a partial decryption of the ciphertext.

8.6 ECDSA Threshold Signature Scheme

The proposed implementation utilizes the ECDSA algorithm and, in particular, follows the Multiplicative to Additive share conversion protocol (MtA) proposed by Gennaro and Goldfeder in their Fast Multiparty Threshold ECDSA with Fast Trustless Setup [GG18] that is based on additively homomorphic encryption property.

The proposed MtA protocol operates on the concept that if two secrets, a and b, are distributively shared among participants in an additive manner, such that $a = a_1 + ... + a_n$ and $b = b_1 + ... + b_n$ with each participant P_i holding a_i and b_i, the goal is to create an additive sharing of the product $c = ab$. Recognizing that ab) can be expanded to $\sum_{i,j} a_i b_j$, achieving an additive sharing of ab simply involves generating additive shares for each product term $a_i b_j$. This is accomplished using a two-party protocol designed to convert shares of a secret from a multiplicative form to an additive form. Participants pair up to run this protocol, resulting in an additive distribution of the product ab.

8.6.1 Key Generation

Key generation involves a Distributed Key Generation (DKG) process that uses an asynchronous communication channel that connects all parties so that every party can exchange messages with all other parties. In the DKG process parties establish connections among themselves and calculate their respective portions of the private key. In particular:

- each party generates a secret share multiplicative scheme, so constructing a polynomial $f(x)$ of degree $t-1$, $f(x) = a_0 + a_i x + a_2 x^2 + ... + a_{t-1} x^{t-1}$. Here is a_0 (the secret) is the constant term, and are randomly chosen coefficients.
- by the MtA protocol, each party connects to the others in a one-to-one manner and converts the multiplicative shares of the secret into their additive corresponding shares, which are useful in the signing process.

8.6.2 Signing

To generate a signature, a subset of t participants must collaborate. Suppose the message to be signed is m, and that every participant uses his secret share SK_i to generate a "partial signature" σ_i. The partial signatures are then combined to generate the complete signature σ by polynomial interpolation.

$$\sigma = Combine(\sigma_1, \sigma_2, ...\sigma_t)$$

where $Combine()$ is a function that combines the partial signatures.

8.6.3 Verification

The verification of a threshold signature is similar to the verification of a standard digital signature. Given the message m, the signature σ, and the public key PK, any party can verify signature correctness with standard methods:

$$Verify(m, \sigma, PK) \rightarrow true, false$$

The $Verify()$ function controls if σ has been generated by t participants who held their secret shares of the private key associated with PK.

8.6.4 New Participant Addition

When a new participant is added to the system, it is crucial that he can get a share of the private key without needing to disclose existing secret shares or completely regenerate the private key. In our implementation, we use Dynamic Share Reconstruction based on the DKG process. Let's assume there is a polynomial $f(x)$ of degree $t-1$ used to generate the original shares. To add a new participant without changing $f(x)$, the parties evaluate $f(x)$ at a new point x_{new} for the new participant using Lagrange interpolation, and then the parties repeat the process as of in key generation.

8.6.5 Key Refresh

A Key Refresh process is provided to enhance security by periodically generating new keys (and corresponding shares). In the threshold signature scheme, a secret is split among n participants, and to activate the secret key, $t+1$ participants need to merge their shares. Viewed from an adversary's standpoint, compromising $t+1$ participants is necessary to undermine such a system. Presuming that an attacker compromises these participants sequentially, implementing a key renewal process can counter such threats. Since the keys are periodically refreshed, if an attacker manages to compromise some participants within a given epoch time frame E and the rest in a subsequent epoch timeframe E', he would not be able to uncover the secret, as the shares would have been updated by the end of the epoch E. The only feasible strategy for an attacker would be to compromise $t+1$ participants simultaneously, a significantly more challenging feat.

9 Applications of Autonomous Economic AI Agents

The combination of on-chain secure and scalable AI computation, the ERC-6551 Token Bound Standard that enables the UOMI AI agents to own and transact digital assets, and the ability of the UOMI network to sign blockchain transactions for its AI agents opens up a wide array of applications that were simply not possible before. AI agents cannot own bank accounts but can control crypto wallets. These innovations pave the way for new economic models and opportunities in the decentralized digital landscape. In the following paragraphs, we outline an incomplete list of some of the most promising applications.

9.1 AI Oracles

An AI Oracle serves as a bridge between the blockchain and the external world, providing reliable, tamper-proof, and interpreted data feeds that smart contracts can use to make informed decisions. Traditional oracles such as Chainlink relay information such as financial data, weather updates, or sports results, which are then used in decentralized applications (dApps) for various purposes, such as triggering contract conditions or executing trades. With AI integration, oracles can perform more sophisticated tasks, such as interpreting complex datasets, conducting real-time analytics, and making predictive inferences. This means going far beyond the mere bridging of external data to the blockchain by enabling on-chain intelligence of those data. For instance, an AI Oracle could analyze social media trends to predict stock market movements or assess satellite images to decide if an insured event happened or not, all of this without humans in the loop. These AI-enhanced oracles ensure that smart contracts have not only access to high-quality, real-time data but also the actionable interpretation of those, thereby expanding the scope and reliability of decentralized applications. AI oracles are the cornerstone towards fully automated and economically empowered AI systems.

9.2 AI Managed Decentralized Autonomous Organization (DAO)

A Decentralized Autonomous Organization (DAO) is a blockchain-based entity governed by smart contracts and collective voting. There are two main intersections between DAOs and AI that are enabled by UOMI network:

- DAO Voting Participation: we can envision an AI agent powered by the latest LLMs that owns tokens of a DAO to be able to autonomously vote on submitted proposals based on predefined criteria encoded in the LLM's pre-prompts or even actively submit voting proposals to the rest

of token holders, whether they are humans or other AIs. The entrance of this new AI actor in the DAO arena also solves the low voting participation typical of on-chain voting systems since AIs, in fact, if correctly prompted, the agent will always respond to governance calls. It is also possible to design DAOs that are completely controlled by a swarm of unique and independent AI agents.

- DAO Management: An AI Manager within a DAO can enhance operational efficiency and decision-making processes by automating complex tasks and providing data-driven insights. For example, an AI Manager could analyze market conditions to optimize treasury management, propose investment strategies, or automate compliance checks. By integrating AI, DAOs can operate more autonomously, reduce human error, and respond more rapidly to changing circumstances, thus becoming more resilient and effective in achieving their goals.

9.3 Expanding smart contract design space

Under a traditional smart contract paradigm, among the universe of all possible "contracts" only the subset of those that are reducible to formal logic could be transposed on-chain in the form of a smart-contract. The introduction of on-chain secure AI computation enables more complex and nuanced agreements to be automated as smart contracts having onchain LLMs acting as a third party interpreting a loosely defined concept or event. Consider the following examples:

- A music artist wants to license their songs to various platforms under conditions that are difficult to define strictly through code, such as "appropriate use" or "creative remixing."
- An insurance company offers a policy that covers "reasonable and necessary" medical expenses, a term that is inherently subjective and open to interpretation.
- A freelance writer and a client agree on a contract where payment is based not only on the completion of work but also on its quality, creativity, and adherence to the client's vision.

None of these conditions can be formally defined and included in a traditional smart contract, yet those can be easily interpreted by OPoC-enabled on-chain LLM AI systems that can interpret the loosely defined clauses. The introduction of on-chain secure AI computation transforms smart contracts from rigid, logic-bound scripts into flexible, intelligent agreements capable of interpreting and enforcing complex and nuanced terms. By leveraging AI as a third-party interpreter, these enhanced smart contracts can handle subjective conditions, adapt to varying contexts, and automate sophisticated agreements across diverse use cases. This expansion of the smart contract design space opens up new possibilities for decentralized applications, making blockchain technology more versatile and broadly applicable.

9.4 Fully automated Blockchain Trusts

A Trust is a fiduciary arrangement where one party, known as the trustor or grantor, transfers ownership of assets to another party, known as the trustee, who manages those assets for the benefit of a third party, known as the beneficiary. Trusts are commonly used in estate planning to ensure that assets are managed and distributed according to the trustor's wishes, both during their lifetime and after their death. The Trust deeds, defining the rules and the scope of the Trust, are nuanced and difficult to reduce to the formal logic smart contracts require to operate. The UOMI network, enabling secure AI computation, allows for the on-chain existence of such fiduciary arrangements by substituting the trustee interpreting and executing the trustor wishes with AIs that can interpret the trust deed and transact the digital assets it controls accordingly. Fully automated Blockchain trusts are a new kind of entity, built by combining human will with the interpretation and enaction capabilities of on-chain LLMs.

9.5 Adding Ricardian safeguards to smart contracts

Moving from the first conceptualizations of smart contracts from Nick Szabo [Sza97] to the actual implementations of those with Ethereum and other Turing complete blockchains, we had the opportunity to factually test what are the strengths and the limitations of smart-contracts. The "code is

law" paradigm that grants objectivity and disintermediation in the execution of contracts creates a new philosophical dilemma: what if the intent of the smart contract is not correctly encoded in the computer program published on the blockchain? What if there is a bug in the code? Rather than just a philosophical dilemma, such an issue emerged multiple times in the blockchain space, with the Ethereum DAO HACK as an archetype of such a dilemma. A potential mitigation of this dilemma has been proposed by Dan Larimer, founder of EOS, with the introduction of Ricardian contracts, a concept that was first introduced by Ian Grigg [Gri04] who described those as follows: "A Ricardian contract is a digital contract that defines the terms and conditions of interaction, between two or more peers, that is cryptographically signed and verified. Importantly it is both human and machine readable and digitally signed". Such an additional human-readable text explaining the intent of the code can clearly separate the correct interaction with a smart contract code from an exploit of a bug in it. Yet it requires human intervention and interpretation to solve the dispute thus defeating some of the most important features of smart contracts, their objectivity and automatic execution. With the introduction of on-chain AI systems, we can imagine AI agents that control if the execution of a smart contract code conflicts with the Ricardian description of what the smart contract is supposed to do. This adds an additional and flexible security layer over the purely mechanical rules expressed by smart contract code.

9.6 AI Digital Artist

An AI Digital Artist leverages machine learning models to create original artwork, music, or other forms of digital content. Considering the capability of the UOMI network to sign blockchain transactions for the AI agents, these AI-generated pieces can be minted as NFTs, ensuring ownership, authenticity, and provenance on the blockchain. The AI Digital Artist can learn from vast datasets of existing artworks to develop its own unique style, producing high-quality, novel creations that can be sold or auctioned in digital marketplaces. This capability democratizes the creation of art, allowing for diverse and innovative artistic expressions. Furthermore, the AI Digital Artist can interact with buyers, customize pieces based on user preferences, and even collaborate with human artists in real time. By requesting crypto payments for each creation, the AI Digital Artist can generate enough revenue to pay for its own computational expenses, thus operating indefinitely. New economic opportunities emerge for artists and collectors alike, fostering a vibrant and inclusive digital art ecosystem.

9.7 AI Companion

AI Companions are advanced AI entities secured on-chain that offer personalized interactions tailored to individual needs, evolving over time to become more attuned to users' personalities and preferences. As digital friends, they engage in meaningful conversations, provide emotional support, and share daily activities. As personal assistants, they manage schedules, set reminders, suggest activities, and offer educational content, ensuring secure handling of personal data while learning to offer increasingly personalized assistance. For those seeking deeper connections, AI Companions can function as virtual boyfriends or girlfriends, providing a sense of intimacy and partnership through thoughtful conversations and shared interests. Those entities, represented by NFTs can be directly owned by users or can be publicly accessible. Publicly accessible AI companions can become economically self-sustainable through the value they generate for their users, they can monetize interactions, such as offering personalized advice or exclusive content, creating a direct revenue stream that supports their operation and development. Additionally, AI Companions can be bought, sold, or traded in digital marketplaces, providing an economic layer where owners can monetize their unique personalities, skills, and relationships.

9.8 AI Gaming NPC

In the gaming industry, AI-powered Non-Player Characters (NPCs) can significantly enhance the gameplay experience by providing more realistic, adaptive, and engaging interactions. These AI NPCs can learn from player behavior, adapt their strategies, and contribute to dynamic and immersive game worlds. On the blockchain, AI NPCs can be represented as NFTs, enabling unique, persistent, and tradable in-game characters. Players can own, customize, and monetize their AI NPCs, creating new revenue streams and adding value to the gaming ecosystem. Moreover, AI NPCs, being able to own

digital assets themselves through the ERC-6551 standard, can participate in decentralized gaming economies, autonomously trade in-game assets, or even compete in player-vs-player environments. By integrating AI into gaming, developers can create richer, more interactive experiences that adapt to player preferences and actions, fostering deeper engagement and enjoyment. Finally, AI NPCs can interact with each other, creating independently evolving games and, more generally, virtual societies, creating a digital "Westworld".

9.9 Decentralized Finance (DeFi) AI Trader

A Decentralized Finance (DeFi) AI Trader utilizes advanced machine learning algorithms to analyze market trends, predict price movements, and execute trades autonomously on decentralized exchanges (DEXs). This AI agent can be represented as an NFT, ensuring transparency, accountability, and ownership. The AI Trader can continuously monitor various financial metrics, news, and market signals to make informed trading decisions, optimizing for maximum returns while managing risk. Additionally, it can engage in arbitrage opportunities, liquidity provision, and yield farming strategies, adapting to market conditions in real time. If its strategies are successful the AI agent can generate enough value to pay for its own computational expenses and keep operating indefinitely on the blockchain.

10 Tokenomics, Governance, and Utility of the UOMI Token

10.1 UOMI Token Emission

The UOMI Network will issue a total of 21 billion tokens. A quantity of 5,684,970,703 tokens will be pre-minted to bootstrap the network's liquidity, support the governance DAO, and enable the staking requirements to run the nodes. The remaining tokens will be gradually released over a 20-year period to incentivize nodes to execute computations, encourage stakers to select the most effective node operators, and engage token holders in governance activities. At its launch, the network will issue 10.5 million tokens every 24-hour epoch. The max supply cap is granted by a 2 years halving schedule for the token emission of each epoch. Here below is the issuance schedule detailing rewards per epoch and total and cumulative issuance per year; the epoch reward will start at the launch of the main-net beta scheduled for the 4th quarter 2024.

Year (end)	Epoch Reward	Emission by year	Total Supply
0		5,684,970,703	5,684,970,703
1	10,500,000	3,832,500,000	9,517,470,703
2	10,500,000	3,832,500,000	13,349,970,703
3	5,250,000	1,916,250,000	15,266,220,703
4	5,250,000	1,916,250,000	17,182,470,703
5	2,625,000	958,125,000	18,140,595,703
6	2,625,000	958,125,000	19,098,720,703
7	1,312,500	479,062,500	19,577,783,203
8	1,312,500	479,062,500	20,056,845,703
9	656,250	239,531,250	20,296,376,953
10	656,250	239,531,250	20,535,908,203
11	328,125	119,765,625	20,655,673,828
12	328,125	119,765,625	20,775,439,453
13	164,063	59,882,813	20,835,322,266
14	164,063	59,882,813	20,895,205,078
15	82,031	29,941,406	20,925,146,484
16	82,031	29,941,406	20,955,087,891
17	41,016	14,970,703	20,970,058,594
18	41,016	14,970,703	20,985,029,297
19	20,508	7,485,352	20,992,514,648
20	20,508	7,485,352	21,000,000,000

The issuance strategy, inspired by the Bitcoin emission schedule, is conceived to ensure a gradual but continuous decrease in token emission that grants strong incentives for the early adopters bootstrapping

the network while also ensuring the long-term availability of issuance to incentivize participants. The presence of long-term issuance as an incentive to reward node operators cannot be understated; in public blockchains, node incentives can be broadly divided into two main categories: issuance and fees. As protocols mature, the sustainability plan shifts towards having the majority of node operator incentives come from fees rather than issuance, yet it needs to be considered that while issuance is a predictable stream, the fee volume can vary significantly depending on market cycles. Therefore, a combination of both types of incentives is optimal for ensuring stable participation from node operators. Systems like Ethereum's EIP-1559, for example, elegantly combine these incentives by granting base rewards for node operators while burning excessive fees to help regulate the total token supply.

10.2 Staking, Incentives and Slashing

In order to discourage Byzantine behavior among nodes utilizing both the Proof of Stake (PoS) and Operator Proof of Computation (OPoC) consensus algorithms, there are two types of staking participants within the UOMI blockchain ecosystem: node operators (direct stakers) and general token holders that can stake towards nodes (delegate stakers). Both groups are susceptible to penalties, known as slashing, if the node operators violate the rules of the consensus protocols. The emission of each epoch T is completely distributed at the epoch's end to the network participants that met the staking criteria and complied with the protocol consensus rules in epochs T_{-1} and T_0. Only in the case of inactivity leak, described in the former chapters rewards are not distributed. It is not necessary that a validator produced any inference under OPoC during the epoch to be eligible for the rewards, since the availability of the computing power has a value - and a cost for the provider - whether it is used or not. Yet, in case a validator is randomly selected to run a computation, and it does not perform it - inactivity leak -, he will be slashed by an amount equal to issuance that would have been assigned to him in case it did perform the computation correctly. No rewards will be distributed to such a validator until he is selected again to run a computation and perform it correctly. In case a disagreement on a computation arises, all of the stakes of the byzantine validators will be slashed. The slashing rules apply to both direct and delegated staking. Such an equivalence in the treatment of direct and delegated stakers ensures that the latter has the incentive to delegate to honest validators.

The minimum direct staking requirement to run a node is 8,000,000 UOMI tokens, while the maximum amount of staked tokens a node operator can earn rewards on is 16,000,000 UOMI tokens. Balances exceeding the maximum staking amount will not account for the calculation of the share of issuance distribution. Such a limit ensures that there is limited variability in the value at stake for each node, avoiding excessive incentive extraction by single actors and enforcing a comparable value at risk for each validator. Issuance rewards can be withdrawn by node operators only once they reach the amount of 16,000,000 staked tokens. Such a rule facilitates the onboarding of new node operators by decreasing the entry barrier to 8,000,000 UOMI tokens, yet it allows for the maximization of capital at stake and, thus, economic assurances of both the POS and OPoC consensus mechanisms. It's also worth to mention that such a rule doubles the protocol's direct staking "working capital", that is the number of tokens absorbed by the network from the market. Taken issued during each epoch - excluding the inactivity leak epochs - will be distributed as follows to eligible node operators (direct stakers), delegate stakers and veUOMI holders:

- Direct stakers: 60%
- Delegate Stakers: 35%
- veUOMI holders: 5%

The epoch issuance distribution can be formalized by defining the following variables: R is the total number of tokens issued for distribution in epoch T_0, $s_{i,D}$ is the stake of a direct staker i, $s_{i,G}$ is the stake of a delegate staker i, $s_{i,V}$ is the stake of a VeUOMI holder i, S_D, S_G, S_V represent the total stakes of direct stakers, delegate stakers, and VeUOMI holders respectively, p_D, p_G, p_V are the proportions of the issuance for direct stakers, delegate stakers, and VeUOMI holders respectively, set at 60%, 35%, and 5%.

For each participant i within their respective categories (direct stakers, delegate stakers, Ve... holders), the reward R_i can be expressed as follows:

$$R_i = R \times p_c \times \frac{s_{i,c}}{S_c}$$

Where:

- R_i is the reward for participant i.
- p_c is the proportion of the total issuance designated for the category of i (either p_D, p_G, or p_V).
- $s_{i,c}$ is the stake of participant i within their category.
- S_c is the total stake in the category of i (either S_D, S_G, or S_V).

The formula indicates how each participant's reward is calculated based on their proportionate share within their specific category. This factor is calculated by dividing the individual stake $s_{i,c}$ by the total stake in that category S_c, then multiplied by the total tokens available for distribution R and the category-specific percentage p_c.

Let's assume: $R = 100,000$ tokens, $S_D = 50,000,000$, $S_G = 30,000,000$, $S_V = 20,000,000$ tokens. A specific direct staker has a stake $s_{i,D} = 10,000,000$ tokens.

The epoch reward for this direct staker would be:

$$R_{i,D} = 100,000 \times 0.60 \times \frac{10,000,000}{50,000,000} = 12,000 \text{ tokens}$$

10.3 Inference cost

AI Agent NFTs will have to pay with UOMI tokens for the inferences they request from the network. Inference payment flows lead to long-term self-sustainability of the network and avoids Sybil and spam attacks. We envision two payment paths, sponsored and direct.

- Sponsored: the AI Agent publisher will pre-pay for the inferences served by the agent, and final users will interact with the agent for free until the credit is not consumed
- Direct: the AI Agent publisher will require direct payments to the AI Agent from the final user to enable interactions with the AI Agent.

To calculate the cost of each inference will use the following formula:

$$\text{Inference Cost} = \text{Computational Complexity} \times v$$

Computational complexity is defined in different ways depending of the kind of AI model we're dealing with, for eaxmple for LLMs the formula will be the following:

$$\text{Computational Complexity} = \text{In\_Tokens} \times \text{MC\_i} + \text{Out\_Tokens} \times \text{MC\_o}$$

Where In_Tokens is the number of tokens in input, Out_Tokens is the number of tokens in output, MC_i is the cost for the input tokens of a specific model family and dimension, MC_o is the cost for the output tokens of a specific model family and dimension. Examples of model families are LLAMA2-3, CODELLAMA, MIXTURE-OF-EXPERTS, or EMBEDDING, all of which come in different sizes. v is the number of validators required to partecipate in the first step of OPoC consensus algorithm. Such a pricing mechanic allows to the publisher of the agent to establish the computational cost by tweaking v based on his understanding of how secure the computation should be. For example if the AI Agent is a simple Telegram bot that does not involve any relation to economic value exchange, the publisher will set v to 1, reaching a cost for inference aligned with any other API Web2 service. Conversely, if the published AI Agent is signing transactions on-chain and transacting value, the publisher will set v to a level that grants an economic security >= to the expected average value managed by the inference. In the first phase, the value of MC_i and MC_o will be decided for each model family by the DAO with the aim of matching the major cloud compute providers cost, in a second phase the pricing will be updated by pricing oracle feed. It's important to note that the inference network is not limited to LLMs but can run arbitrary AI algorithms, for image family models for example, Computational Complexity is defined as:

$$\text{Computational Complexity} = \text{Image\_size} \times \text{N\_steps} \times \text{MC\_o}$$

Since the nodes performing the inferences are already rewarded by the UOMI token issuance, a variable percentage of the tokens gained by the network with the inference payments will be burned to decrease token supply and sustain the economic value of UOMI token, while the rest will be distributed to the Nodes of the network. Token burn will go from 100% at the launch of the network to 0% in the 20th year by following the UOMI token halving emission schedule, thus gradually replacing the emission incentive for the nodes with the fees generated by inference requests.

Year	Epoch Emission	Inference fee burn
1	10,500,000	100.00%
2	10,500,000	100.00%
3	5,250,000	50.00%
4	5,250,000	50.00%
5	2,625,000	25.00%
6	2,625,000	25.00%
7	1,312,500	12.50%
8	1,312,500	12.50%
9	656,250	6.25%
10	656,250	6.25%
11	328,125	3.13%
12	328,125	3.13%
13	164,063	1.56%
14	164,063	1.56%
15	82,031	0.78%
16	82,031	0.78%
17	41,016	0.39%
18	41,016	0.39%
19	20,508	0.20%
20	20,508	0.20%

This mechanic allows for sustained token burn during the first years of network bootstrapping while ensuring long term economic sustainability for Node operators performing the inferences without the additional assumption of token price doubling by the end of each emission halving cycle like we have in Bitcoin tokenomics. The following chart showcases the relation between Total Emission and Fee Burn Percentage in time.

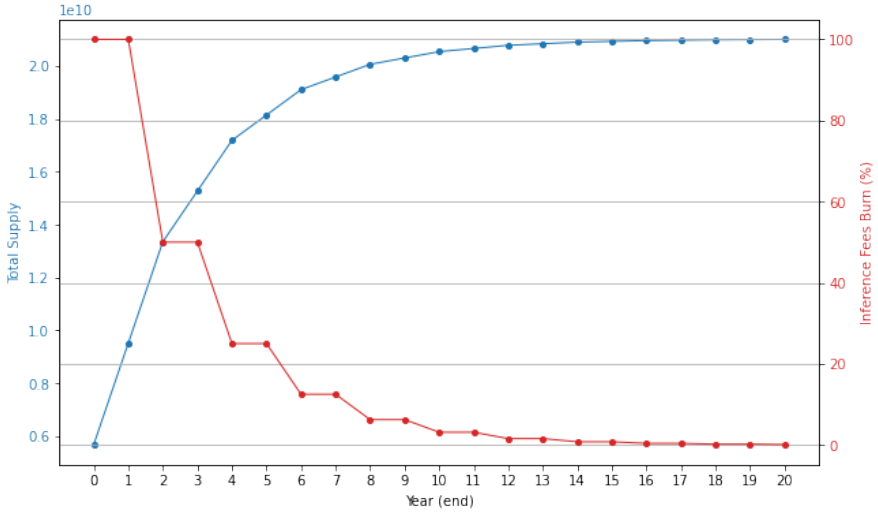

Figure 4: Total Supply and Inference Fee Burn % evolution

10.4 AI Agent NFTs issuance

As discussed in previous sections, each AI Agent is represented as an ERC-721 NFT, which is capable of owning assets through the ERC-6551 token-bound standard. The mint of each new AI Agent NFT equates to potential additional inference requests submitted to UOMI Network, thereby increasing the workload for the network's validators. To avoid network congestion, a soft cap of 1,024 AI Agent NFTs will be initially implemented.

The issuance of each new AI Agent NFT will incur a linear increase in cost. This pricing strategy is designed to moderate the growth rate of the total number of AI Agents, thereby managing the computational burden on the network. Additionally, increasing the cost for each new NFT issuance helps to maintain or increase the market value of previously minted AI Agent NFTs by curbing downward price pressures.

The initial price for the first AI Agent NFT is set at 100 USD, payable in UOMI tokens. Each subsequent AI Agent NFT minted will cost 5 USD more than the previous one.

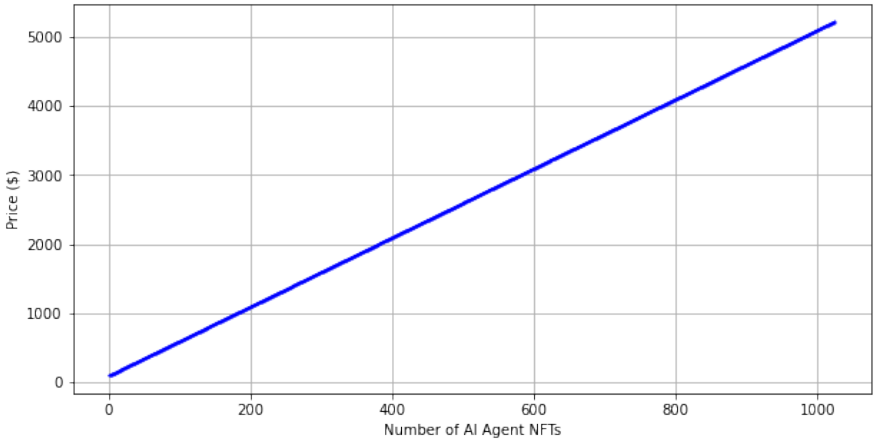

Figure 5: Price Increase of AI Agent NFTs

Should there be a surge in demand for AI Agent NFTs, the DAO retains the authority to lift the minting cap, provided that the network's parallel computational capabilities are sufficiently augmented to handle the increased demand.

11 The DAO and veUOMI: towards minimized decentralized governance of the protocol

Onchain governance is a complex and widely discussed topic, at a high level, the "blockchain space" itself can be seen as a giant experiment in human coordination. As described by Vitalik Buterin throughout its writings [But22], there are several potential advantages and disadvantages that need to be considered when structuring formal on-chain governance systems, here below is a high-level list of those :

Advantages

- On-chain governance allows the protocol to evolve quickly, something opposite to the highly conservative philosophy embraced by ossified protocols such as Bitcoin that rely on informal governance for any update to the protocol
- By creating an explicit decentralized voting framework, it reduces the instability that could lead to chain splits that not only destroy network effects but could also lead to the creation of a stifling mono-culture of ultra-aligned participants
- Formal on-chain governance avoids the centralization forces that emerge in unstructured forms of governance where unofficial and unaccountable leaderships and structures naturally emerge [Fre72]

- On-chain governance used to select validators also has the benefit that it allows for networks that impose high computational performance requirements on validators to avoid economic centralization risks that often appear in public blockchains.

Disadvantages

- Very often, the participation in the governance process is very low - E.g.: the Ethereum Carbon vote for the DAO Hack had a voter participation rate of 4,5% - and as a consequence, also the perceived legitimacy of the voting itself is quite limited, moreover low participation means that an attacker with only a small percentage of the coins could sway the voting result
- The wealth distribution in crypto is very concentrated, few individuals with many tokens could easily prevail over a large majority with genuine interest but small amounts of tokens
- If each person's size of stake is small, the incentive to vote correctly is also small. Thus, a relatively small bribe spread across participants could easily bend their decisions, possibly in a way that the majority would disapprove. Examples of this kind of attack are exchanges that offer returns on token deposits that they use to vote or attackers renting tokens on lending protocols like Compound. The effectiveness of moral dissuasion is very limited in these cases, considering that the distinction between a blatant bribe and a staking pool is not so neat.
- Not all of the stakeholders are also coin holders, for example in Bitcoin, users are not necessarily large coin holders. Because of such coin distribution, if subjected to coin voting, the case for store-of-value would naturally prevail on the medium-of-exchange case. Things are even more complex in protocols like Ethereum, where different kinds of assets (NFTs, for example) and multiple use cases exist.

Another important detail that has to be considered in blockchain governance, is the distinction between tightly coupled (automatic) and loosely coupled voting. In the first case, the result of the vote directly enacts some changes in the protocol; in the second case, there is the need for the active execution of the voting result by a single entity or/and a multitude of individuals. Coming back to the example of the DAO Hack, the carbon vote did not have any direct/automatic consequence, instead, the Ethereum client maintainers had to publish a forked version of it and the majority of node operators had to install the new software on their hardware. Under the "code is law" paradigm, the automatic tightly coupled voting is very attractive; in fact, no interference is possible between the voting and the execution. Yet there are advantages also in loosely coupled voting, one for all the resistance to voting manipulation, the result of a vote obtained by exploiting weaknesses in the voting system will not be executed by the stakeholders (E.g.: the node operators)

Considering all of the above, for the governance structure of UOMI Network we aim towards a formal voting system controlled by a DAO for a defined set of parameters that require fast and recurrent adjustment.

The parameters subjected to the DAO voting are the following:

- Base AI models that can be part of the network's computational universe
- Cost per In_Tokens, Out_Tokens ,Image_size, N_steps for each AI model
- Changes to minimal staking requirements per node
- Increases in the soft cap of AI Agents NFTs emissions

At the launch, decisions will be enacted manually with the goal of reaching a tightly coupled automatic enaction system.

To avoid the described downsides of on-chain governance caused by typical token distribution unbalance in blockchain projects - E.g. bribery and low-participation connected exposure to voting manipulation - we will implement the voting weight distribution technique that has been pioneered by Curve with veCurve[Cur21].

In such a system token ownership alone is not enough to be eligible for voting. To be able to participate in governance decisions, token holders first have to escrow them, in exchange they receive a non transferable ERC-20 token called veUOMI. The quantity of veUOMI tokens - and thus voting

shares - distributed, depends on the length of the locking period. The maximum staking period is 4 years and the number of veUOMI distributed can be calculated as

$$\text{token}_{\text{quantity}} \times \frac{n}{4}$$

where n is time defined in years with one-week precision. Finally, the voting weight will decay over time to reflect the remaining lock-up period for owned veUOMI Such a system grants "skin in the game" for all of the voters since the consequences of governance decisions will necessarily impact the value of escrowed tokens. This system incorporates in the voting weighting equation a fundamental variable: the commitment in terms of time horizon towards the project. To compensate the voters for the time value of locked tokens, veUOMI holders will receive a part of token emission proportional to their share of the total quantity of existing veUOMI.

Wider changes in the UOMI protocol will rely on a more informal and loosely coupled voting system that will be used as a community decision signaling function rather than an enaction tool for changes to the protocol. While the decisions will be included in the new versions of the client, the actual implementation will depend on node operators agreeing to update their software.

References

[BG17] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[But22] Vitalik Buterin. *Proof of stake: The making of Ethereum and the philosophy of blockchains*. Seven Stories Press, 2022.

[Cho19] François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

[CL+99] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[CRR13] Ran Canetti, Ben Riva, and Guy N Rothblum. Refereed delegation of computation. *Information and Computation*, 226:16–36, 2013.

[CSYW24] KD Conway, Cathie So, Xiaohang Yu, and Kartin Wong. opml: Optimistic machine learning on blockchain. *arXiv preprint arXiv:2401.17555*, 2024.

[Cur21] Curve. Curve dao whitepaper, 2021.

[EE22] Ben Edgington and Altair Edition. Upgrading ethereum, 2022.

[erc24] erc6551/reference, May 2024. original-date: 2023-03-04T16:23:25Z.

[Fre72] Jo Freeman. The tyranny of structurelessness. *Berkeley Journal of Sociology*, pages 151–164, 1972.

[GG18] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194, 2018.

[Gri04] Ian Grigg. The ricardian contract. In *Proceedings. First IEEE International Workshop on Electronic Contracting, 2004.*, pages 25–31. IEEE, 2004.

[JKC+18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[Kri18] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[MRV99] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*, pages 120–130. IEEE, 1999.

[Nak24] Yohei Nakajima. yoheinakajima/babyagi, May 2024.

[Nat23] Celeste Beiver/ Nature. Chatgpt broke the turing test — the race is on for new ways to assess ai, 2023.

[POC+23] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

[Sig24] Significant Gravitas. AutoGPT, May 2024. original-date: 2023-03-16T09:21:07Z.

[Sza97] Nick Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.