The AO Protocol: A Decentralized Open-Access Supercomputer

Sam Williams
sam@arweave.org

Ivan Morozov
ivan@autonomous.finance

Tom Wilson
tom@hyper.io

Tyler Hall
tyler@hyper.io

Vincent Juliano
vince@arweave.org

Alberto Navarro
alberto@arweave.org

DRAFT-8
June 13, 2024

Abstract

This paper delineates the protocol of the AO computer, a decentralized computing system inspired by the actor-oriented paradigm. It establishes a single system image capable of supporting numerous parallel processes without the constraints typical of current decentralized computation models, emphasizing network verifiability and minimized trust requirements. The architecture of AO is extremely modular, facilitating seamless integration with existing smart contract platforms and allowing customization across computational resources, virtual machines, security mechanics, and payment mechanisms. Key functionalities include unrestricted resource utilization for hosted processes, direct integration with Arweave's data storage capabilities, autonomous activation of contracts, and a comprehensive message-passing layer for inter-process coordination. This protocol focuses on providing a terse overview of the computer's mechanics, in order to accompany its formal protocol specification.

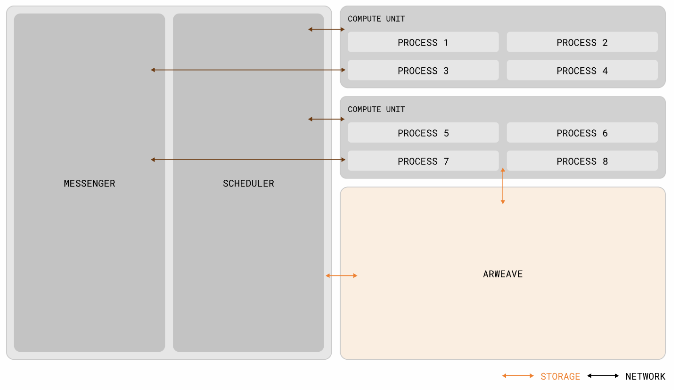

Figure 1: The AO computer architecture takes a modular approach to its construction: Each logical responsibility is split into an appropriate subnet, the participants of which each engage in a peer-to-peer market for the provision of their services.

1 Introduction

The AO computer is the actor oriented[13] machine that emerges from the network of nodes that adhere to its core data protocol, running on the Arweave[8] network. This document gives a brief introduction to the protocol and its functionality, as well as its technical details, such that builders can create new implementations and services that integrate with it.

The AO computer is a single, unified computing environment, a Single System Image[14], hosted on a heterogeneous set of nodes in a distributed network. AO is designed to offer an environment in which an arbitrary number of parallel processes can be resident, coordinating through an open message passing layer. This message passing standard connects the machine's independently operating processes together into a 'web' – in the same way that websites operate on independent servers but are conjoined into a cohesive, unified experience via hyperlinks.

Unlike existing decentralized compute systems, AO is capable of supporting the operation of computation without protocol-enforced limitations on size and form, while also maintaining the verifiability (and thus, trust minimization) of the network itself. Further, AO's distributed and modular architecture allows existing smart contract platforms to easily 'plug in' to the network, acting as a single process which can send and receive messages from any other process.

Instead of enforcing one set of choices upon all users of the computing environment, AO is built in a modular form: Allowing users to choose which virtual machines, sequencing models, message passing security guarantees, and payment options work best for them. This modular environment is then unified by the eventual settlement of all messages – each sharing the same format – onto Arweave's decentralized data layer. This modularity creates a unified computing environment suiting an extremely wide set of workloads, in which every process can easily transfer messages and cooperate.

AO's core objective is to enable trustless and cooperating compute services without any practical bounds on scale. This allows for a radically new design space of applications that were not previously possible: Blending the benefits

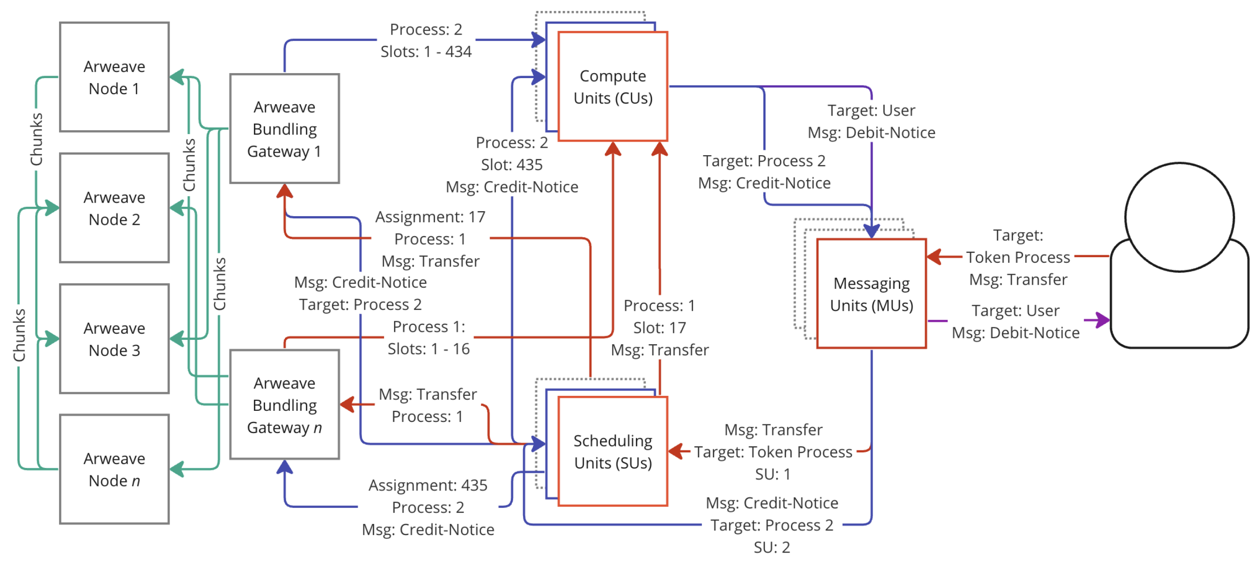

Figure 2: The inter-node communication flow while processing a typical request from a user (in this a transfer). Red lines indicate the path of information between nodes as a result of the originating message, while the magenta and blue lines follow the path of subsequent messages. Green lines trace an example distribution of the content of these messages – ensuring their long-term data availability – between nodes on the Arweave network.

of smart contract applications (services without requiring trust in anything but code), and traditional compute environments (Amazon EC2, etc.).

2 Core Functionality

`aos`—a decentralized operating system for AO—allows developers to launch command-line processes that function like smart contracts within its decentralized network. This process is similar to starting a server on a cloud service, but with decentralization and trustless computation as key advantages. These processes operate without being confined to any particular location, enabling seamless user interactions across the network. The outcome is a 'Single System Image'—a unified, global computing platform that transcends physical and scalability limits, collectively used by all participants. Essentially, AO forms a vast, scalable computer where users can interact with any process, promoting a highly collaborative ecosystem.

For users, AO represents a shared computer on which they can execute multiple processes. These processes are not confined to any particular servers or under the dominion of any single individual or group. Once activated, these processes deliver their services with cryptographic security, ensuring unbiased and perpetual operation. This design empowers users with the ability to rely on services that maintain their rights consistently over time, fostering a trustable environment for interaction with the system.

When compared to existing decentralized and distributed computation systems, the AO protocol offers a number of novel mechanisms and features. In this section we will run through some of the core benefits it provides in turn.

2.1 Arbitrary numbers of processes ('contracts') running in parallel

In AO, applications are built of any number of communicating processes. Inspired by the original actor model[19] and Erlang[16], AO does not allow processes to share memory between one another, but does allow them to coordinate via a native message-passing standard. Each of these processes can then be operated at the full speed of the computing resources that are available, without interfering with one another. By focusing on message-passing AO enables scaling mechanics that are far more similar to traditional web2/distributed systems environments, than traditional smart contracts.

2.2 Unbounded resource utilization in processes

Building on the lazily-evaluated architecture of the original versions of SmartWeave[25] and LazyLedger[5] later known as Celestia[11], nodes in the AO network do not need to perform any compute at all in order to reach consensus about program state transitions. State is implied 'holographically' by the Arweave-hosted log of messages to the process. Compute costs are then delegated to users who can either calculate their own states, or request execution by nodes of their choosing.

2.3 Access Arweave, a native unbounded hard drive

AO processes can seamlessly load and execute data of any size directly into their memory and write back to the network. This setup eliminates the typical resource constraints and enables fully parallel execution, dramatically expanding the possibilities for application development beyond the limits of traditional smart contract platforms. Consequently, it opens the door to sophisticated applications requiring extensive data handling and computational resources, such as machine learning tasks and high-compute autonomous agents.

2.4 Autonomously activating contracts

In traditional smart contract environments (like Ethereum, Solana, Polygon, etc.), contracts 'wake up' to perform compute at the request of a user transaction. This creates an environment in which programs are not 'live' unless a user interacts with them, lessening the scope of applications that can be built on top. AO removes this limitation by allowing contracts to have scheduled 'cron' interactions that automatically wake them up and execute compute at set intervals. Any user, or indeed the process itself, can pay a node to 'subscribe' to a process in order to trigger the evaluation of the compute at the appropriate frequency.

2.5 Modular architecture supporting extensions

AO's core architecture is an open data protocol that anyone can build an implementation of. Everything – the sequencers, message passing relayers, and even the virtual machine of the system—can be swapped out and extended at will. This flexibility will allow the existing smart contracting systems in the Arweave ecosystem (Warp, Ever, Mem, et al) to plug into AO and be able to send and receive messages from the unified network. This will also allow all of these smart contracting systems to share some of the same infrastructure and tooling, making for a more coherent experience of compute on Arweave.

3 Architecture Overview

Now that we have established the purpose and features of the AO computer network, we will consider the core components of its construction. The fundamental elements of AO are as follows:

3.1 Processes

Processes are the network's unit of computation. Processes are represented by a log of interacting *messages* stored on Arweave, as well as an initialization data item. Processes define their required computing environment (its VM, scheduler, memory requires, and necessary extensions) in their initialization. While processes are represented at the consensus level in this way, they also imply a state which can be calculated by *computing units* that satisfy the requirements and choose to execute the process. As well as receiving messages from user wallets, processes are also forwarded messages from other processes via *messenger units*. The developers of processes are given free choice as to how

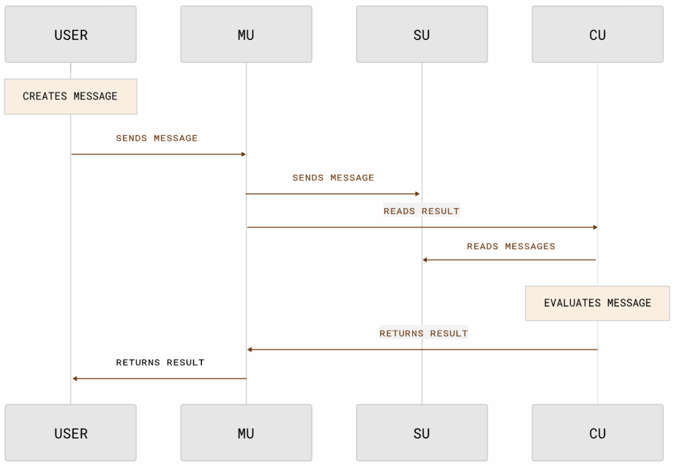

Figure 3: Nodes from applicable subnets cooperate in the AO computer protocol in order to fulfil user interactions. Each member of the respective subnets competes in a scale-free marketplace with other nodes to offer highest quality of service (including minimized fees and latencies) for end users.

to determine the trustworthiness of these messages, as described in the below sections.

3.2 Messages

Every interaction with a process in AO is represented by a message. At their core, messages are ANS-104[6] compliant data items. Users and processes (via their *outboxes* and *messenger units*) can send messages to other processes on the network by way of *scheduler units*. Messages in AO have semantics between that of UDP[20] and TCP[21] packets: Delivery is guaranteed to occur only once, but if the message is never forwarded by a messenger unit—or the recipient never actually processes it—then its delivery will not occur.

3.3 Scheduler Units (SUs)

Scheduler Units are responsible for the single assignment of atomically incrementing slot numberings to the messages that are sent to a process. After assignment, schedulers are required to ensure that data is uploaded to Arweave and thus made permanently available for others to access. Processes have free choice of their preferred sequencer, which can be implemented in a variety of ways: Decentralized, centralized, or even user-hosted.

3.4 Compute Units (CUs)

Compute Units are nodes that users and *messenger units* can use in order to calculate the state of processes in AO. While SUs are obligated to sequence the messages of processes they have accepted, no CU is required to calculate the state of a process. This gives rise to a peer-to-peer market for computation, where CUs offer the service of resolving process state in competition with one another—trading off price, the computation requirements of the process, and other parameters. Once computation of a state is completed, the CU will return to the caller a signed attestation of the *output* (logs, outboxes, and requests to spawn other processes) of the resolution of a specific message. CUs may also generate and publish signed state attestations that other nodes can load—optionally for a UDL[17] specified fee.

3.5 Messenger Units (MUs)

Messenger Units are nodes that relay messages around the AO network according to a process called *pushing*. In essence, when MUs push a message around the system they send it to the appropriate SU for a process, then coordinate with a CU in order to calculate the output of the interaction, and then repeat the process recursively for any resulting outbox messages. This process continues until there are no more messages to push. Users and processes can also pay a MU to *subscribe* to a process, pushing any messages that result from its timed *cron* interactions. Processes can also optionally label a message as a *cast*—leading the MU to send the message to its SU, but not listen for a response. In this way, AO is able to provide a vibrant environment that gives users and processes maximal choice—VM, payment method, scheduler type, messaging security, and more—without requiring consensus on costly computation itself.

4 Related Work

There are no direct analogies to draw upon that describe what AO is and the experience of using it. There are, however, many adjacent projects and networks that can be used to contrast with AO in order to elucidate its properties. In this section we discuss each in turn.

4.1 The Actor Model

The Actor Model, introduced by *Carl Hewitt, Peter Bishop*, and *Richard Steiger* in their paper, *A Universal Modular Actor Formalism for Artificial Intelligence*[19], serves as a foundational framework for understanding and implementing concurrency in computer systems. This model posits that the fundamental unit of computation is the “actor,” an entity that can make local decisions, create more actors, send messages, and determine how to respond to messages it receives. This approach to system design and programming facilitates the creation of distributed, highly concurrent, and scalable applications.

4.2 Erlang

AO is largely inspired by the Erlang computing environment and its programming language. Erlang is an implementation of the actor model which offers extremely lightweight processes, handled by schedulers in the runtime, in order to enable efficient utilization of massively parallel systems (machines and networks with many physical threads). These capabilities give rise to a 'process-oriented' form of programming, in which the developer naturally splits their computation into many cooperating and parallel components in order to achieve their goal. While Erlang is not extremely well known amongst mainstream computing circles, it is used in a significant number of environments where high performance is a necessity: Telephony switches[16], instant messaging services like WhatsApp[15], etc.

The AO computer derives its process-oriented approach from Erlang directly. Erlang offers clue evidence that an environment in which distributed computation is achieved through processes that pass messages but do not share memory can be highly efficient. AO applies this approach to the domain of smart contracts, while also offering a single system image for an Erlang-like environment for the first time.

4.3 Smart Contracting Platforms (Ex. Ethereum)

Ethereum is a decentralized computing network in which all users share memory and a single thread of execution. Originally based on an idea of adding Turing Complete computation to a blockchain[2], Ethereum morphed into a project to build a 'world computer'[18]. Upon launch, Ethereum was able to demonstrate the power of trustless computation of arbitrary code – without the production of an independent blockchain network – for the first time. While the network gained immense traction with users and developers, the core network's throughput has not improved since it launched in 2015.

Instead of attempting to scale the base network past the processing capacity of a single, small thread of execution, the Ethereum ecosystem has pivoted to a 'rollup-centric' roadmap[1]. This approach to scaling focuses on supporting additional 'rollup' networks that inherit some[24] of the properties of Ethereum, but not all of them. At the time of writing, there are 14 rollups in the Ethereum ecosystem with more than $100 million of total value represented in their programs. Each of these 14 rollups represents another single thread (a 'process' in AO terms) of computation that can be performed in parallel. By building from the ground-up to focus on parallel execution rather than shared memory, AO offers a completely novel architecture that supports an arbitrary number of independent processes, while maintaining the ability for programs to be decentralized and trustless.

4.4 Decentralized Compute Marketplaces

In traditional smart contract platforms such as Ethereum, a shared-thread architecture restricts each user to only executing small computational tasks. This inherent limitation constrains both the complexity and scalability of operations on the network, thereby impeding the potential for more computationally intensive smart contracts.

4.4.1 Decentralized Large-Scale Computing Networks

Several networks, such as Akash[4], aim to facilitate large-scale computing in a decentralized context. Unlike platforms that prioritize verifiable and reproducible computations, Akash and similar networks provide a decentralized marketplace for container hosting services. This model supports the execution of traditional, non-deterministic programs on x86 architecture physical machines, though it compromises the ability to create trustless services characteristic of smart contracts.

4.4.2 Advancements in Virtual Machine Technologies

AO allows developers to choose their preferred Virtual Machine (VM), with the initial reference implementation focusing on WebAssembly (WASM). WASM containers in AO can manage up to 4 GB of memory—a limit expected to increase with the adoption of WASM64, thereby enabling prolonged duration computations. The rich compilation tools within the WASM ecosystem support a diverse range of programs, exemplified by recent uses in executing LLM transformer models[10], speech recognition[9], and compute-heavy image manipulation software like Photoshop[3] in web browsers.

4.4.3 Holographic State Mechanism in AO

Despite its substantial computational capacities, AO maintains traditional smart contract execution capabilities due to its holographic state mechanism. Rather than achieving consensus on the state of the computation itself, AO ensures that *logs* of interactions are recorded and accessible on Arweave. This setup projects a 'hologram' of the state, meaning that while the state may not have been computed by any participant yet, it is guaranteed to always produce the same outputs when computed. Furthermore, the holographic state system, powered by message logs on Arweave, allows AO processes to react to implied messages on a timed basis, thereby facilitating proactive actions.

Coupled with its holographic state mechanism, AO also offers a distributed network of *Compute Units* that provide cryptographically signed attestations about the results of computations. These compute nodes engage in a competitive market, which serves to reduce the costs associated with resolving the holographic state, thereby enhancing efficiency for users.

4.5 Peer-to-Peer VM hosting

Urbit[26] is a peer-to-peer computation system with some similarities to AO. By focusing on the transfer and availability of interaction logs, Urbit offers a distributed compute environment where 'servers' can be ported from one physical host to another. During the transition, the log of interactions with the hosted computation can be executed to recalculate the current state. Additionally, Urbit processes can send messages to each other to facilitate communication.

Unlike AO, Urbit does not achieve decentralized consensus on its interaction logs. Practically, this means there is no canonical agreement or guaranteed availability of its 'rollups' — consequently, the state of its processes remains uncertain. In this respect, Urbit shares characteristics with Akash and other decentralized compute marketplaces but also allows for the verifiable migration of computation from one host to another, *if* the host is agreeable to the transition. AO advances this model by ensuring that logs of messages to a process are accessible via Scheduler Units (SUs), which upload them to Arweave. This enforcement of log availability decentralizes user processes—no longer confined to a single computation node—allowing their state to be resolved by a distributed network of Compute Units (CUs) in real-time. This architectural difference provides AO with the necessary attributes for deploying trustless smart contracts and supporting a vast number of processes. This capability is further enhanced by the fact that processes can be holographically represented; their message logs are permanently accessible, even without any CUs currently attached to execute them.

4.6 Internet Computer Protocol

The Internet Computer Protocol (ICP) shares some objectives with AO, such as creating a decentralized verifiable computation environment. However, the mechanisms and architectural choices within ICP diverge significantly from those of AO, leading to different operational paradigms and potential limitations.

ICP employs a single Byzantine Fault Tolerant (BFT) mechanism across its 'subnets', a design choice that mandates consensus on the *results* of computations. This necessitates that every node within a subnet execute every step of each computation, inherently limiting the amount of computation that can be feasibly performed due to scalability constraints. Furthermore, ICP adopts a monolithic protocol structure, enforcing uniform consensus and execution parameters across all resident containers.

By contrast, AO employs a modular approach, where different network responsibilities are segmented into subcomponents with flexible parameters. This design philosophy extends, for example, to allowing processes within AO to choose their virtual machines and security parameters — defining an environment tailored to their specific needs rather than conforming to a one-size-fits-all model.

Additionally, AO's core design focuses on achieving consensus around the *inputs* to processes, rather than exclusively on the outputs. This approach permits processes in AO to operate for any desired length of time, enhancing the system's adaptability and application scope. Moreover, AO's governance is minimized, resembling Bitcoin's model, where the network operates in a truly permissionless manner without the intervention of any controlling organization. In contrast, ICP employs a governance-heavy approach where a DAO can revoke participation rights and has the authority to de-platform any container it deems necessary. This centralized control is akin to a public-company operated by its shareholders, potentially leading to discriminatory practices against certain protocol uses.

Furthermore, ICP's security model is based on node operators undergoing 'KYC' processes with the DAO, lacking protocol-enforced economic incentives that guarantee execution fidelity. This contrasts with AO's economic model, which is designed to foster a competitive, open market that naturally aligns node behavior with network health through economic incentives.

5 Network Security

The AO protocol adopts a modular approach to its technical architecture, imposing minimal specific requirements on its resident processes. This principle extends to AO's security mechanisms as well. The core components of the network—its data protocol, sub-unit role division, and integration with Arweave—offer a framework for building secure computations without mandating a single approach for all resident applications. This flexibility enables the network to adapt to a wide variety of use cases, supporting its mission to provide a universal protocol for decentralized computation. In this section, we explore the construction of AO's modular network security architecture, the formal process model that underpins it, and two exemplar security processes that will be available in the live network: "AO-Sec Origin' and SIV. Afterwards, we will present the economic fundamentals that set the base for a market around this security model.

5.1 General Overview of the Security Model

In addition to allowing varied security mechanisms to be layered on top of its data protocol and data replication system (Arweave), AO supports the stacking of security processes on top of one another, enabling users to freely combine their benefits and trade-offs. Two such mechanisms are: 'AO-Sec Origin', which provides rehypothecatable collateralized message passing, and SIV, a mechanism for incentivizing Sybil-resistant attestation sets for network actions (see sections 5.6 and 5.7 respectively). While these systems can be used independently, stacking them together yields both Sybil-resistant and collateralized message transmission. To further showcase how modular security enhance the network, we now describe how security can be passed along processes by means of staking economic value.

5.1.1 Hierarchical Security

Processes in AO operate independently with deterministic verifiability of their individual states. As previously described, processes coordinate through a system of message passing, relayed by Messaging Units (MUs). Each of these relayed messages comes with a cryptographically verifiable signature produced by the MU. Each process has the autonomy to decide how to respond to messages from different signers, allowing them to choose their own security model based on their needs. This setup enables various security mechanisms with different trade-offs (like latency, cost, and efficiency). Further, each process's state transitions can be calculated independently, without relying on messages from other processes, allowing the network to scale without needing to fully verify all processes for any single message validation.

Security mechanisms in the AO network are based on a universal principle: Attestations on the outcomes of message interactions with processes should be cryptographically validated and economically secured. The AO data protocol provides cryptographic validation, while the AO-Sec Origin security process ensures economic security. To achieve this, the AO-Sec process allows any network participant to:

- 'Stake' a token representation of economic value in the process itself. While staked, this collateral may be subjected to votes that may lead to its 'slashing' – removal from the ownership of the 'staker' – upon the agreement of other stakers.
- Deposit economic value into a 'sub-staking' process, granting it authority over the funds, allowing it to slash or return them according to its own rules. self-administered rules.

AS an additional feature, The AO-Sec Origin process also offers 'back-stop' liveness and Scheduler Unit (SU) failure recovery mechanisms, detailed in section 5.6.1.

5.1.2 Economic Security

The AO network requires a native token to support all safety mechanisms, implementing economic security for processes that rely on the protocol. As a consequence, the AO token is introduced into the system to underpin the network's 'AO-Sec Origin' security process. It serves as a liquid and common unit of value for additional economic mechanisms layered above it, as described in the section above. In order to achieve this in the most neutral way possible, the token's launch mechanics have been designed to closely resemble Bitcoin's monetary policy and have been optimized to effectively bootstrap the network's economic security layer. Thus, AO has no preordained token allocations for any type of network participants. Instead, every token is distributed proportionately to the value of assets introduced into the system, with all movements generating demand for economically secured message passing. These mechanisms are described in detail in Section 7.

With the introduction of the AO token, an economic framework emerges from which any number of downstream security mechanisms can be constructed. These security mechanics reside in their own AO processes, which users can deposit their AO tokens into in order to participate. Rather than offering a single security mechanic that is globally applied to all users, choice is instead granted to both staked service operators and clients to find a mutually acceptable means of interaction. As a result, a market is created in which varied security mechanisms compete for acceptance, while a unified token offers a mutually admissible liquid unit of economic value to underpin them.

The versatility of this security model will allow for the possibility to leverage new technologies, such as ZK proofs, to validate the integrity of messages within the network without the need of the core AO protocol to undergo any changes. However, economic security will continue to be essential for providing message ordering and data availability. Only significant breakthroughs in distributed and cryptographic systems, which are not anticipated in the near future, could jeopardize the integrity of this economic model.

In summary, when considering the totality of all of its components, AO's technical and security mechanics offer a radically novel techno-capital construction. The network provides its users:

1. A trustless computing environment that supports an arbitrarily sized workload;
2. Efficient markets for the provision of each of the services inside that environment;
3. Customizable, user-defined security mechanisms that allow many varied workloads with differing requirements to exist in parallel and interoperate with one another;
4. A novel, and more coherent token economic model for a network that does not experience block-space scarcity.

In the following section, we set to formalize the general risk model for AO processes, which will allow us to analyze how different AO (sub-)staking security modules address specific protocol needs. The two main modules that dissipate these risks are 'AO-Sec Origin' and 'SIV' – a sub-staking process of AO-Sec which offers a simple and fast AO attestation consensus protocol.

5.2 Formal Security Model of the AO Computer

To develop a coherent model of threat vectors in the AO network, we need to define its core components: processes, messages, and attestors. With this foundation, we can then examine the key roles in the protocol, their behaviors, and the specific threat models associated with each.

5.2.1 Processes

Let P_i represent the i^{th} process.
Define $P_i = (Log_i, Init_i, Env_i)$, where:

Log_i is the ordered sequence of all messages for P_i.

$Init_i$ is the initialization data for P_i.

$Sched_i$ is the scheduler for P_i.

Env_i is the computing environment for P_i.

Notably, as the ao data protocol focuses on providing a universal format for decentralized and verifiable computation, it does not enforce a specific virtual machine, nor any associated parameters. Subsequently, when a developer creates a new process on ao, they can specify all of the parameters necessary for units in the system to deterministically execute it. These parameters are added as tags on the spawning data item and may include (but are not limited to):

- The maximum amount of memory that the process should be able to use.
- The maximum number of operations (optionally weighted, according to the specification of the virtual machine) that the process may consume while evaluating a single message.
- Any extensions to the virtual machine that the process requires (access to a virtualized local file system, hardware-optimized encryption instructions, etc.), as defined by the virtual machine specification.

The state of P_i at a given time step, $S(P_i)$, is determined by:

$$S(P_i) = F(Log_i, Env_i)$$

where F is a function, defined by Env_i, computing the state based on the message log.

The outbox of new messages to be sent to related processes as a result of a message is described as follows:

$$Outbox_m = F(Log_i, Env_i, m)$$

where m refers to the originating message.

5.2.2 Messages

Let M_{ij} represent the j^{th} message in P_i. M_{ij} is represented as an ANS-104 compliant data item. The delivery status $D(M_{ij})$ can be represented as:

$$D(M_{ij}) = \begin{cases} 1 & \text{if delivered} \\ 0 & \text{else} \end{cases}$$

The AO data protocol employs at-most-once delivery semantics, as detailed in [22], atop which additional guarantees are provided by the maintenance of message logs on Arweave through its data persistence protocol. These guarantees ensure that undelivered messages that result from P_i may always been delivered later by re-computing $Outbox(P_i)$ from its message log on Arweave.

5.2.3 Attestations

Let S_U denote the *stake* for a unit performing attestations U, representing the value of locked tokens committed by the unit to ensure economic security for action it is involved in. The stake is defined as:

S_U =tokens committed by S in a staking process.

S_P =the active staking process of the attestor.

Once staked, the tokens S_U for any A may be subject to *slashing* as a result of malicious behavior, in accordance with the ruleset of its active (sub-)staking process (S_P).

5.3 Scheduler Units

Upon receiving a message m, a *SU*, denoted as SU_{P_i} for process P_i, performs the following operations:

1. **Assignment:** SU_{P_i} assigns m a unique incremental nonce, n, reflecting the order of receipt relative to other messages within the same process. This assignment is formalized as:

$$a(m) = (m, n, \sigma(SU_{P_i}, m, n))$$

where $\sigma(SU_{P_i}, m, n)$ denotes the cryptographic signature of SU_{P_i} over the message m and its nonce n.

2. **Persistence:** The signed assignment, along with the message, is persisted onto the Arweave data layer, ensuring its availability and integrity within the network.

5.3.1 Mechanics

Secure staking processes should SU_{P_i}, denoted as $S_{SU_{P_i}}$, should be subject to slashing by an acceptable and active staking process S_P under the following conditions:

1. If SU_{P_i} fails to perform the assignment for m or maliciously drops m, $S_{SU_{P_i}}$ will be slashed to penalize the non-compliance.

$$\neg a(m) \Rightarrow Slash(S_{SU_{P_i}})$$

2. If SU_{P_i} performs the assignment for m but fails to persist the signed assignment and message onto the Arweave data layer, resulting in a 'gap' in the log for P_i, $S_{SU_{P_i}}$ will also be slashed.

$$\neg Persist(a(m)) \lor \neg Persist(m) \Rightarrow Slash(S_{SU_{P_i}})$$

3. Assigning a slot more than once with the same nonce to different messages:

$$\exists m_1, m_2; m_1 \neq m_2 \land A(m_1)_n = A(m_2)_n \Rightarrow Slash(S_{SU_{P_i}})$$

5.4 Compute Units

Compute units execute the virtual machine (defined by Env_i) function λ for P_i on given a message:

$$\lambda(P_i, m_j) = \langle \Phi_{P_i}, Outbox_j, Attest_j \rangle$$

where Φ_{P_i} is the new process state, $Outbox_j$ is the set of any resulting outbound messages, and $Attest_j$ is a signed attestation of the computation.

5.4.1 Mechanics

Should $Attest_j$ be determined to be incorrect by other parties in the staking process, they should have the authority to commence a slashing operation against S_{CU}. Subsequently, MUs can employ the results of $\lambda(P_i, m_j)$ from a CU, with economic guarantees bounded by S_{CU}, ensuring a secure and reliable framework.

5.5 Messenger Units

Messenger Units (*MUs*) act on behalf of the user in order to move messages between processes in the system. By performing this task, called *pushing*, MUs are able to orchestrate any number of processes in order to perform specific tasks for users.

5.5.1 Mechanics

1. MU_m receives a message m_i from a client or user.
2. The message m_i is then forwarded to the scheduler SU_k for assignment and publication, ensuring it receives a unique slot in the process's ordering.
3. MU_m requests the outbox of a chosen CU_l for any new messages that have been generated as a result of processing $\lambda(P_i, m_i)$.
4. If there are new messages in the outboxes, MU_m takes each new message, signs it, and forwards it to the appropriate SU_k, recursively continuing the process.

The recursion ends when there are no more new messages given by CU_l for all prior messages, signifying the end of the processing cycle for the user's interaction.

$$\text{Push}(MU_m, M) = \begin{cases} \emptyset & \text{if } M = \emptyset \\ \text{Process}(M, MU_m) & \text{otherwise} \end{cases}$$

where;

$$\text{Process}(M, MU) = \begin{matrix} \{\text{SU}_k(\sigma(MU, m)) \mid m \in M\} \\ \cup \{\text{Push}(MU, \text{CU}_l(m)_{\text{out}}) \mid m \in M\} \end{matrix}$$

$\sigma(MU, m)$ returns m signed with MU's private key.

5.5.2 Message Acceptance

Messages are propagated via $Push(MU_m, m)$, either directly by the Messenger Unit MU_m or by external initiators. Upon receipt, processes evaluate these messages and their signatures to decide on subsequent actions: either to engage (α), ignore (ι), or request re-transmission with different security parameters (ρ). These may include using an alternate security subprotocol (a different staking process), or a different stake quantity or time. This protocol empowers processes within the AO network to delineate their security requirements for message interaction, symbolically represented as:

$$\text{Decision}(P_i, m, \sigma) = \begin{cases} \alpha, & \text{if security criteria are met,} \\ \iota, & \text{if it is to be disregarded,} \\ \rho, & \text{if re-transmission is required.} \end{cases}$$

In the event that MU_m is discovered signing an invalid message, stakers within the given staking process should enforce a slashing operation against the stake of MU_m, denoted as S_{MU_m}.

$$\neg MUm \Rightarrow \text{Slash}(S_{MU_m})$$

Moreover, should the invalidity stem from a Compute Unit's (CU) attested result, $\lambda(P_i, m_j)$, MU_m may assert a claim against the CU's stake, S_{CU}, contingent on the staking process's framework. This relationship is defined as:

$$\neg\lambda(P_i, m_j) \Rightarrow Transfer(S_{CU}, S_{MU_m})$$

5.5.3 Stake Aggregation

To securely relay messages across the network, MUs may need to aggregate attestations from multiple CUs to meet the security requirements of the processes. This aggregation ensures that the combined stake is sufficient to uphold the integrity and trustworthiness of the message being transmitted.

The process of stake aggregation operates as follows:

1. A MU identifies the security requirements specified by the process for a given message.

2. The MU then collects attestations from available CUs, each contributing a portion of the required stake.

3. These attestations are aggregated into a single stake bundle, effectively pooling the security guarantees from multiple sources.

4. The aggregated bundle is then relayed through the Scheduling Unit (SU) to the process (operated by a CU).

This bundled approach to stake aggregation allows for a streamlined and efficient transmission of secured messages, ensuring that each message meets the predefined security criteria before it is processed by the recipient CU. Moreover, by utilizing a composite of attestations, the system enhances the resilience and fault tolerance of message handling, distributing the risk among multiple CUs and thereby mitigating potential points of failure.

$$\text{Aggregate}(MU_m, m) = \bigcup_{CU_i \in C} \sigma(CU_i, m)$$

At the discretion of the CU or the process itself, attestations from *specific* CUs may be required.

5.6 The AO-Sec Origin Process

We now examine the function and design of the 'AO-Sec Origin' process, which serves as the backbone of security on the AO network. This foundational process underpins the economic security mechanisms for all subordinate mechanisms within the network, ensuring robust network integrity.

The AO-Sec Origin process acts as the primary custodian and issuer of the network's staking tokens, holding ownership records and user-defined properties regarding all stakable units. It also provides the back-stop security functions for the reliable operation of the network, including staking, slashing, and unstaking of tokens. Finally, AO-Sec Origin facilitates the reassignment of processes in response to failures or breaches of protocol by Scheduler Units (SUs), such as liveness issues or double-signing.

5.6.1 Trustless Scheduling Guarantees

The Ethereum ecosystem has successfully pioneered a novel approach to 'sequencing' (giving unique ordering to, and ensuring availability of) transactions in decentralized networks [1]. Rather than focusing on providing traditional Byzantine Fault-Tolerance (BFT) to every transaction in normal operating conditions (which incurs significant costs – both in performance and economic burden), equivalent safety can be provided by the ability to trustlessly *fallback* to traditional decentralized consensus only when necessary. In this paradigm, users gain the full benefits of decentralization (liveness, censorship resistance, and trustlessness), without the necessity of bearing the cost of multi-party consensus upon sequencing. Concretely, the communication complexity of AO's approach compares to traditional blockchains as follows:

Network	**Best**	**Average**	**Worst**
AO Network	$\Omega(1)$	$O(1)$	$O(n \log n)$
Traditional Blockchain	$\Omega(n \log n)$	$O(n \log n)$	$O(n \log n)$

Table 1: A comparison between the communication complexities of message settlement on AO and traditional networks [12].

Scheduler Units can encounter three primary types of faults, each of which has a different resolution mechanism in the AO-Sec Origin process. Their details are as follows:

1. **Liveness:** In which a processes active SU (SU_p in our formal model) is offline or non-responsive to requests to schedule specific messages (censorship).
   Resolution: At any time, any network participant may raise a challenge on the AO-Sec Origin process for the responsible SU to schedule any message onto Log_i for process P_i. If the SU does not respond to the challenge within the given period (set by the process during initialization), P_i becomes *unhosted*.

2. **Double Assignment:** A malfunctioning or malicious SU may provide two signatures for the same 'slot' for

a process ($\exists m_1, m_2; m_1 \neq m_2 \land A(m_1)_n = A(m_2)_n$), leading to ambiguity in the ordering of messages.
Resolution: Upon acquiring evidence of a double assignment of slots ($A(m_1)_n$ and $A(m_2)_n$), any participant in the network may submit both assignments to the AO-Sec Origin process. This causes P_i to become unhosted. Notably, $(m_1)_n$ (the *first* m_n to be submitted to the process) becomes the selected message at Log_i for P_i. This essentially performs the function of 'fork recovery' in a traditional blockchain, restoring an unambiguous total ordering to the process.

3. **Non-Publication:** Sometimes, due to negligence or malfeasance, the entity scheduling a process might fail to publish a message for which it has provided an assignment ($A(m)_n$). If this happens, a Computation Unit (CU) attempting to calculate the state $S(P_i)$ for a process, as Log_i would be incomplete.
Resolution: Any participant in the network can raise a challenge, requiring that the Scheduling Unit SU_P (or any other interested party) publish the message m_n directly to the AO-Sec Origin process within a specified timeout window. If the challenge fails, the process P_i becomes *unhosted* at Log_{n-1}. Unlike other AO-Sec Origin operations, non-publication challenges may involve the forced publication of large messages, which can be costly. To mitigate this burden, processes have the following two options at their disposal during initialization:
   (a) **Specifying additional non-publication challenge processes:** During launch of a new computation, users may choose to specify an additional process that must be monitored by its corresponding SU in order to respond to non-publication challenges. These processes implement the same interface as the AO-Sec Origin process, operating as sub-staking modules on the network. In the event of SU faults on the additional non-publication challenge process, they too can fallback to the AO-Sec Origin process, due to the *hierarchical* security structure that it provides. SU operators may choose at their discretion to charge greater fees to host a process that has additional non-publication challenge addresses specified, as they must observe the process to check for challenges over time.
   (b) **Allowing stakers to vote on data availability:** Some processes may choose to allow staked members of the AO-Sec Origin process to vote on whether the data is available, rather than forcing publication of the data itself. If the process specifies this option, the cost of forced-publication may be saved (which may be reflected in a lower hosting fee by SUs), but creates exposure for the process to the majority of the stakers' opinions.

Notably, in every resolution scenario aside *3B* (an optional, secondary approach) the process is not exposed to issues of honest or dishonest majorities in the AO-Sec Origin staking committee. The result of this, is that processes that do not choose to be exposed to optional voting mechanics (typical in modern *PoS* networks) have mechanistic trustless operation guarantees that do not require them to have faith in the integrity of the node operators over time.

In order for the AO-Sec Origin process to confer the trustless guarantees listed above, it itself must be hosted on a more traditional blockchain. Because AO's data protocol allows free choice of the type of SU that a process resides on, AO-Sec Origin is able to use Arweave's Byzantine Fault Tolerant (BFT) consensus algorithm [7] as its host. This mechanism functions in the same way as SmartWeave and other, first generation smart contracting techniques on Arweave [25]. By operating in this way, liveness of the AO-Sec Origin process is ensured, thereby extending these properties to all processes within the AO network. These properties even extend to any other sub-staking security processes inside AO: In the event that their SU stops hosting them, these processes can fallback on Arweave's robust BFT consensus. This system allows for rapid transaction processing by these sub-staking processes under normal conditions, with the security of traditional BFT mechanisms available in emergency situations.

5.6.2 Markets for re-hosting processes

In the event of challenge failures in AO-Sec Origin, processes may be moved to an *unhosted* state. In order to trustlessly resolve these scenarios, any willing SU may send an unscheduled (unreplicated, or assigned) 'dry-run' message to the process in order to gain its assent to become the host for the process. In an *unhosted* state, the process is unable to send messages to other processes or change its state (as the dry-run input message to the process itself has not been assigned a slot), but it is able to respond to the caller in expressive ways. It may:

1. Accept the SU's offer to become its host (accepting the fees that the SU quotes for its operation).

2. Ask for further information from the SU. In this scenario, the SU is able to retrieve this information and dry-run the interaction with the process again. Despite the lack of ability to modify its own state during its *unhosted* phase, through this mechanism the process is still able to interact with the full AO environment in order to decide whether to accept the offer.

3. Reject the offer without further request.

Once a SU has produced a message that leads to an affirmative response from the process, they may submit their m to AO-Sec Origin in order to become the valid host (SU_p) for the process. Through this mechanism processes are able to 'negotiate' with would-be hosts and gain the necessary information needed to decide the most appropriate new SU, avoiding any need for 'forced-assignment' by votes.

5.6.3 Sub-Staking and Sub-Ledger Processes

The parallel design of the AO network allows for the implementation of further sub-staking and sub-ledger mechanisms. These 'child' processes enable the deployment of AO tokens across a wide variety of security frameworks and payment systems:

Sub-Staking Processes: These processes offer customizable security configurations to cover the spectrum of diverse needs of network participants. By enabling bespoke security, the network grants participants with high adaptability and robust protection for their processes, ensuring close alignment with their specific requirements.

Sub-Ledgers: Sub-ledgers are processes that enable the parallel execution of payments while holding a token balance in the parent process. They are versatile, capable of extending beyond mere token storage to include a range of functionalities that enhance transaction processing efficiency. As delineated in the AO Token Specification, subledgers facilitate the seamless transfer of tokens between a parent process and its child processes, provided these adhere to the established token interface standards [23]. Trust in the originating module of a sub-ledger allows for users to be represented by the process with the parent token without direct intervention, thereby circumventing potential bottlenecks in the primary process. Additionally, if the module (its compiled code) managing a sub-ledger is considered trustworthy, the balances maintained across these processes may be regarded as fungible. Tokens received from one subledger of a parent (in this case, the AO-Sec Origin process) may be deemed indistinguishable from those received from another. This architecture permits an indefinite expansion of parallel processes, which in turn is capable of supporting an indeterminate volume of simultaneous transactions.

Through this structure, the security properties of the AO-Sec Origin process are transitively applied to custom-built security mechanisms downstream while enabling payments in the network's native token at arbitrary throughput rates.

5.7 Exemplar Sub-staking Process: SIV

The SIV sub-protocol enhances the foundational security features of AO's staked message transfer system by introducing an additional layer of Sybil-resistance, ensuring the possibility to stake in single-assignment mode, and bolstering data availability guarantees. Particularly, SIV integrates a lightweight, low-latency consensus mechanism through result attestations, tailored for deployment in specific scenarios within the AO network as per requests by either a process or a user. These scenarios include:

1. Consensus among Sybil-resistant staked parties on messages resulting from network computations.

2. Enhanced assurances against the rehypothecation or 're-staking' of security collateral within the fraud detection time frame specified by a message recipient.

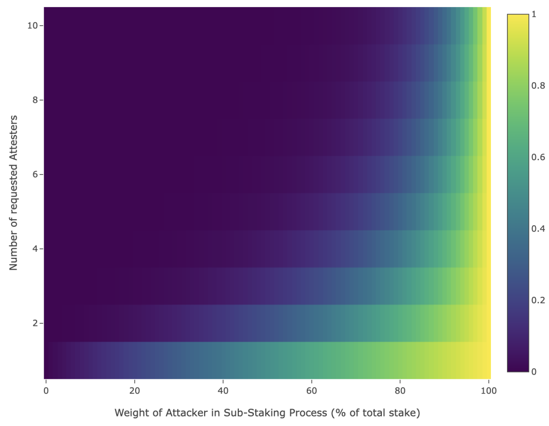

Figure 4: Utilizing the SIV sub-staking process significantly reduces the probability of deceptive actions by an attacker with each additional attestor, while the costs for clients increase linearly with more stake-time access.

3. Improved guarantees on the single-assignment of slots for processes.

The operational framework of SIV includes a deterministic ordered set of attestors assigned to audit other stakers' activities. Clients requiring staked operations can mandate the inclusion of SIV, specifying the necessary number of attestors' signatures for result validation. This flexible attestation requirement allows clients to effectively balance their needs for security, cost, and latency.

SIV's consensus mechanism is notably streamlined, allowing for users to choose between complete consensus on results when needed, or otherwise operate with partial consensus to increase efficiency. In other words, clients can define the exact number of participants needed for a given action, thereby controlling the cost and latency impacts of consensus on outputs. This strategic choice empowers users to achieve complete consensus on results, while normally operating with partial consensus which remains computationally efficient at $O(1)$ complexity, contrasting with the $O(n)$ complexity typical of full blockchain networks.

SIV primarily enhances security by providing Sybil-resistant attestations. These complement AO's economic safeguards, such as the over-collateralization of messages based on their potential economic impact. By achieving consensus on messages and actions through stochastically validated attestations from a dynamically sized set of attestors, the likelihood of a staker deceiving a client decreases exponentially with each additional attestor involved, as illustrated in Figure 4.

The probability that all attestors are Sybils controlled by an attacker is mathematically represented as:

$$P(s,n) = \left(\frac{s}{100}\right)^n$$

where P denotes the probability of all attestors being Sybil, s represents the stake percentage controlled by the attacker, and n indicates the number of attestors requested by the client.

The operational sequence of a SIV sub-staking process is detailed as follows:

1. Any party can trustlessly initiate a new process with the SIV module by specifying operational parameters such as memory and instruction limits, supported VM environments, minimum stake time periods, and fraud penalties.

2. Post-initialization, AO service operators meeting the SIV criteria can partake by aligning their tokens with the SIV instance via the AO-Sec Origin process.

3. With the closure of each epoch, the process entropy is renewed, integrating new stakers and reshuffling attestor sequences to ensure comprehensive coverage of all active stakers.

4. Stakers aggregate necessary attestations to fulfill client requests, optimizing response times by paralleling attestor responses.

5. Clients validate these attestations against the current attestor set, ensuring all responses are duly signed and relevant.

6. Stakers can issue a termination notice at any time, allowing them to withdraw from attestation obligations and transfer their duties to the next staker in the sequence.

5.7.1 Ensuring Attestor Liveness

In order to address any unresponsive or non-compliant auditors, SIV incorporates a mechanism to automatically remove them from the committee after a preset duration. Should an auditor fail to respond to a staker's challenge within a preset duration, they are automatically removed from the committee. This ensures continued responsiveness and reliability within the network. However, to prevent potential abuse of this system, stakers initiating a challenge must pay a fee. This fee, calculated as a stake-weighted average of all stakers' proposed rates, is burnt to regulate the token supply and align staker incentives towards maintaining a robust, fair-priced attestation mechanism.

6 Economic Model

The typical economic model of blockchain networks like Bitcoin, Ethereum, and Solana, revolves around the concept of buying access to scarce block space, with security being subsidized as a byproduct. Users pay transaction fees to incentivize miners or validators to include their transactions in the blockchain. However, this model inherently depends on the scarcity of block space to drive fee revenue, which in turn funds network security.

In the context of Bitcoin's security architecture, which is fundamentally underpinned by block rewards and transaction fees, consider a hypothetical scenario wherein block rewards are eliminated and transaction throughput is assumed to be infinitely scalable. Under these conditions, the scarcity of block space would effectively be nullified, leading to minimal transaction fees. Consequently, the economic incentives for network participants to maintain security would be significantly reduced, thereby increasing the vulnerability of transactions to potential security threats.

Solana exemplifies this theoretical model in practice, illustrating that as network scalability increases, fee revenues correspondingly diminish. In the absence of substantial transaction fees, the principal source of security funding is derived from block rewards. These rewards essentially function as a tax on token holders, manifesting either as operational overhead for those electing to stake their tokens personally, or as a gradual dilution of their proportional ownership within the network for those who abstain from staking.

Earlier, we presented the need for an AO token as a unified representation of economic value to support security mechanisms within the network. Below, we discuss how the market for security emerges in AO, how participants can benefit from dynamic stake exclusivity and how this market reaches equilibrium by calibrating its parameters.

6.1 Establishing a Market for Security

In contrast to the examples above, AO introduces a novel approach where users purchase the specific level of security required for each message they send. This model allows users to "insure" their messages to the level of security deemed necessary by their counterparts, facilitating a tailored and efficient allocation of security resources. This direct relationship between message security and user expenditure obviates the need for subsidizing security through block rewards or collective fee-bargaining mechanisms.

Furthermore, AO's model creates a competitive market for access to the network's stake that underpins security. Since security is purchased on a per-message basis, a dynamic marketplace for staking emerges, where the price of security is determined by supply and demand rather than fixed network rules. This market-driven approach promotes efficient pricing and allocation of security resources, providing robust security tailored to the actual needs of users and avoiding the one-size-fits-all model of traditional network mechanics.

This economic model not only enhances efficiency but also aligns the incentives of all network participants by directly correlating the cost of security with its consumption. This alignment potentially reduces the overall operational costs of the network.

6.2 Stake-Exclusivity Periods

An integral feature of AO's security framework is the implementation of 'stake-exclusivity' periods. This security mechanism enables the recipient of a message to designate a specific time window during which the stake used to secure the message is exclusively reserved—preventing it from being ‘double spent’—for that particular message’s transmission. Throughout this exclusivity period, the staked collateral is locked, rendering it unavailable for other uses, thereby ensuring its availability for potential slashing if discrepancies in the message are later identified.

This feature significantly enhances the trustworthiness of the transmission process by allowing recipients to set a ”stake-exclusivity” period that matches the security requirements of their specific transactions. Stakeholders can customize security measures based on the perceived risk and value of the message. For critical or valuable messages, extended exclusivity periods can be set, providing sufficient time for necessary challenges and verification by concerned parties. This arrangement not only bolsters security but also strengthens the integrity of the message verification process.

6.3 Time-Value of Stake

In the AO model, each message that is passed requires the user to compensate for the ’time value of stake’ — the opportunity cost of locking (single-assignment) of capital for a specific duration to secure a message. This mechanism plays a critical role in determining the pricing dynamics within the AO system.

6.3.1 Economically Rational Security Pricing

Consider a user who wishes to insure a message worth \$1 million with a stake-exclusivity period of 15 minutes. The cost of insuring this message—paying for the time value of the stake—can be modeled as a function of the expected annual return rate desired by the stakers. For instance, if stakers expect a 10% annual return on their engaged capital, the cost of securing this message can be derived from the pro-rata share of this expected return over the exclusivity period.

The formula for calculating the price, P, of securing a message for a period E in a given unit (minutes, seconds, etc.), given the total insured amount I and the annual expected return rate r (expressed as a decimal), is defined as follows:

$$P = I \cdot \left(\frac{R}{T_{\text{annual}}} \right) \cdot E$$

where:

I is the total collateral provided for the insurance,
R is the annual expected return rate of the staker,
E is the stake-exclusivity period.

For a message with collateral of \$1 million with a 15-minute exclusivity period and a 8% expected return rate:

$$C = \$1,000,000 \cdot \left(\frac{0.08}{525,600} \right) \cdot 15 = \$2.28.$$

This calculation indicates that the time-value price for a 15-minute exclusivity on a \$1 million stake, assuming an 8% annual return, is \$2.28. Message recipients are empowered to select their preferred over-collateralization ratio and stake-exclusivity duration, effectively balancing risk against the cost of service access. This flexibility is a testament to the network’s underlying economic principle: offering a customizable environment where users and services can adjust parameters—such as stake amounts and timeframes—to meet the demands of their specific operations.

6.3.2 Market Dynamics and Equilibrium

The equilibrium in the AO staked messaging market is influenced by the interaction between the demand for security (driven by the value and stake-exclusivity time of messages) and the supply of stake capital (influenced by staker return expectations). The demand for security stake can be expressed as a function of the quantity of transactions Q and their average value $\bar{V}$, adjusted by the average stake-exclusivity period $\bar{E}$:

$$D = Q \cdot \bar{V} \cdot \frac{\bar{E}}{T_{\text{annual}}}$$

where T_{annual} is the total number of the units in a year for which stake exclusivity is being considered (for example, 525,600 for minutes). This formula reflects the total demand for stake capital in terms of the average economic value that is active in message security at any given time.

The supply of economic utility from staked capital S, is modeled as the product of the total stake available K and the expected return rate R:

$$S = K \cdot R$$

This equation reflects the total economic value that the staked capital is expected to generate over a year. Here, K represents the volume of AO tokens actively committed to securing the network, and R is the annualized expected return on these staked tokens.

Equilibrium in the staking market $D = S$ is achieved when the demand for security, D, matches this supply of economic utility, S:

$$Q \cdot \bar{V} \cdot \frac{\bar{E}}{T_{\text{annual}}} = K \cdot R$$

Solving for the required return rate R, which balances the supply and demand for staking utility, yields:

$$R = \frac{Q \cdot \bar{V} \cdot \bar{E}}{K \cdot T_{\text{annual}}}$$

This formula calculates the equilibrium return rate R. If this calculated R is higher than the current market rate of return, it indicates a deficiency in staking capital relative to the demand for security. Consequently, more capital will flow into staking, increasing K until the new equilibrium is reached. Conversely, if R is lower than the market rate, it suggests an oversupply of staked capital, leading some stakers to withdraw their funds, thereby reducing K until equilibrium is restored.

6.3.3 Peer-to-Peer Market Dynamics

The AO network's decentralized, peer-to-peer market structure inherently allows nodes to independently set their own fees for staked message passing services, without enforcing global pricing. This flexibility lets them dynamically adjust to market demand and supply changes, fostering competition and boosting responsiveness. Nodes that offer competitive rates and terms naturally attract more users, optimizing their returns and leading to an efficient market equilibrium.

This mechanism promotes market efficiency while laying the foundation for well-defined token valuation metrics. The process of analyzing the volume and value of secured messages, along with competitive return rates, establishes a comprehensive framework for real-time token valuation, which is dependent on the network's perceived security, utility, and demand.

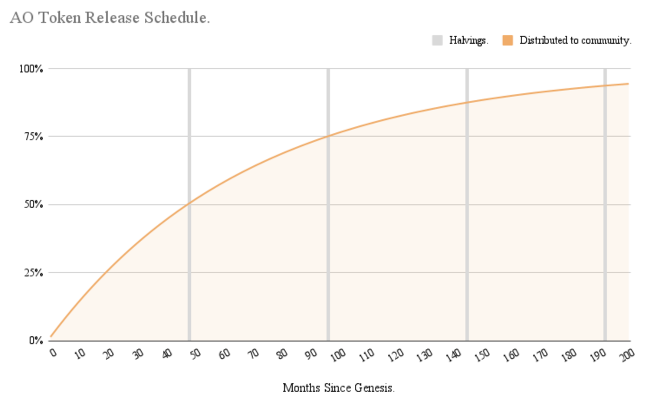

Figure 5: Growth of the AO token supply over time. AO's model follows Bitcoin's, modified to generate a smooth emission curve – with 'halvings' representing a *half-life*, rather than an abrupt change to the emissions per time period.

7 AO Token Minting

Once an efficient market for security on the AO network has been defined, it is essential that the AO monetary policy guarantees that no party has an unfair economic advantage without having vested interests. To achieve this, we now set to outline a minting process that distributes new tokens only to network participants who have a direct economic interest in leveraging the token for its intended purpose of securing the AO network.

7.1 AO supply growth over time

AO will have a total supply of 21 million tokens. New tokens are minted every 5 minutes, accumulating to a monthly rate of 1.425% of the remaining supply. As a result, the circulating supply follows a log curve, with the amount of minted tokens effectively undergoing a halving event every 4 years. In contrast to the Bitcoin network, where the coin reward per block is abruptly cut in half every 4 years, the AO network will always disperse tokens at the same rate, so that the number of tokens that are minted every 5 minutes will diminish continuously as the circulating supply grows over time.

In the absence of pre-mined tokens or other discretionary supply shocks, the circulating supply of AO tokens can be predicted with full certainty at any point in time since genesis (see figure 5).

7.2 Distribution of AO supply

Arweave serves as the structural foundation for the AO network. It not only enables its security mechanisms, but its native token AR allows for messages on AO to be stored and subsequently validated in a truly decentralized manner. For this reason, AR addresses holding a positive balance will be eligible to receive AO for as long as tokens are minted. This minting mechanism is designed to create a natural alignment of interests of AR token holders with the AO network that should foster the mutual success of both networks in the long term.

The minting of AO tokens begun with the launch of the AO testnet on February 27th 2024. After a period of 4 months, users looking to mint AO with assets other than Arweave´s native token AR have the option to deposit and bridge qualified assets into the network. From the moment this functionality is enabled, two thirds of the supply of newly minted AO tokens is distributed to users that hold bridged assets in the AO network. AR holders will continue to receive one third of the ongoing supply. In the case where no qualified assets are bridged into AO, AR holders will continue to receive the totality of new tokens.

This monetary policy heavily rewards those that bring assets into the AO network. Parties that bridge qualified assets will consequentially be entitled to a share of the AO token supply for as long as their assets remain in the network.

Thus, we can express the number of AO tokens minted per wallet U_R as

$$U_R = \begin{cases} \frac{1}{3} \cdot T_R \cdot U_{RAR} + \frac{2}{3} \cdot T_R \cdot U_{RB} & \text{if } T_B > 0 \\ T_R \cdot T_{RAR} & \text{if } T_B = 0 \end{cases}$$

where:

T$_R$ is the total number of AO tokens minted per time period,
T_B is the total number of bridged assets per time period,
U_{RAR} is the per wallet share rate for holders of AR tokens, and
U_{RB} is the per wallet share rate for holders of qualified bridged assets.

With this formula we can now calculate the appropriate reward rates for AR token holders U_{RAR} as well as addresses that own bridged assets U_{RB}.

7.2.1 Reward rate per wallet for holders of AR tokens

The amount of AO tokens awarded for holding AR is determined by the ratio of the AR an address holds compared to the total circulating supply of AR.

Then, the per wallet share rate for holders of AR tokens U_{RAR} can be expressed as

$$U_{RAR} = \frac{U_{AR}}{\sum_{i=1}^{n} U_{AR_i}}$$

where U_{AR} is the number of AR tokens in the user's wallet.

7.2.2 Reward Rate per wallet for bridged assets

The amount of AO tokens awarded for depositing qualified bridged assets into the AO network is determined by the ratio of the volume of the bridged asset times its annual staking yield compared to the the total volume of assets that have been bridged to the AO network. As a result, bridging qualified assets the carry higher native staking yield, all else equal, will lead to a greater wallet share rate for newly minted AO tokens.

The per wallet share rate for any holder of qualified bridged assets U_{RB} can be expressed as

$$U_{RB} = \frac{U_B \cdot NSY_B}{\sum_{i=1}^{n} U_{B_i} \cdot NSY_{B_i}}$$

where:

U_B is the volume of qualified bridged asset B per wallet, and
NSY_B is the native annual staking yield per qualified bridged asset B.

7.2.3 Criteria for Qualified Assets and Bridges

The main criteria that bridged assets must fulfill in order to qualify for payments of newly minted AO tokens are:

1. The asset must be mature, with a diverse and liquid market.
2. The asset type must be yield bearing with robust, battle-tested liquid staking tokens such as stETH.

For this criteria to be fully met, assets must enter AO via bridges that offer the functionality to split the underlying asset from its native yield in order to fund the growth of the network.

Bridges to the AO network consist of two smart contracts, one on the network the assets are bridged from, the native chain, and another one on the AO network itself. The smart contract on the AO network generates derivative tokens that represent the bridged native assets. The derivative tokens are freely transferable on the AO network. Once the yield bearing asset has been deposited into a bridge, the owner is given a representation of that token that is compatible with the AO network token standard, e.g bridged stETH is represented in AO aoETH. Users can then freely move and stake their tokens across the network, while passively minting AO. Assets can be withdrawn from AO by their owners at any time, but resign from then on to the eligibility to mint AO for the portion of the assets that are bridged back to its network of origin.

7.3 Ecosystem Development

The AO minting and distribution design presented above encourages any interested party to move liquidity into the AO network in order to strongly incentivize the growth of its economy. As a direct result of this model, unlike most modern blockchain networks there is no central party with control over a treasury of AO tokens in order to drive ecosystem development and adoption. Since the supply of AO tokens cannot be altered or controlled, AO token holders cannot be directly or indirectly taxed for future enhancements to the network or discretionary initiatives.

In order to support the further development of the AO ecosystem, the network offers two mechanisms of funding for its growth:

1. A Permissionless Ecosystem Funding, and
2. Permaweb Ecosystem Development Guild (PEDG)

The funding rate of these ecosystem growth mechanisms will decrease in accordance with the rate of decay that the AO token mint curve is subject to. As AO token rewards diminish over time, so too will the yield extracted from bridged assets decline in a proportionate fashion.

7.3.1 Permissionless Ecosystem Funding

The Permissionless Ecosystem Funding offers an opportunity for developers to fund their applications and establish thriving profit sharing communities with their users and liquidity providers. Users that bridge their assets into AO will have the liberty to interact with a wide variety of applications. By doing so, they not only provide liquidity to these apps, but also temporarily give the right for their native AO yield to the process – providing it with a stream of funding.

Developers that are able to attract capital for their applications will automatically have the mandate to decide the best use of this long-term source of revenue. Builders may choose to use all of these funds for their application, or share these AO tokens amongst their users. They can even utilize this flow of new AO tokens to fund other relevant applications and services, if they prefer. Liquidity providers, on the other hand, have the freedom to decide the apps and teams they wish to fund based on their personal preferences. In this way, a permissionless. meritocratic, and transparent revenue stream is provided to developers of new applications, freeing them from the need to request grants or investment funding.

7.3.2 Permaweb Ecosystem Development Guild (PEDG)

The PEDG is an alliance of dedicated AO ecosystem organizations and builders that develop, grow, and maintain the infrastructure necessary for the AO network. The PEDG is funded by the native yield generated by bridged assets, while they are in use on the AO network. Rather than funding a single core team, this yield is distributed amongst a diverse set of teams and builders contractually committed to the growth of AO. At the time of AO's token launch, PEDG is composed of 5 ecosystem partners that collaborated on the launch of the AO network, with more to be added as the protocol matures and grows.

8 Conclusion

The AO protocol presents a significantly differentiated design in the realm of decentralized computing by implementing a model based on the actor-oriented paradigm. This approach enables the AO network to operate without the traditional constraints faced by similar systems, primarily through its modular architecture and a flexible security model. These features enable the network to adapt to the diverse needs of its users, creating a more versatile and efficient environment for decentralized applications.

Central to AO's design is its capacity to support an unlimited number of parallel processes. This capability significantly boosts the network's scalability, also allowing for the coexistence of various configurations for bespoke operational requirements. This capability significantly boosts the network's scalability, also allowing for the coexistence of various configurations for bespoke operational requirements.

Moreover, AO's economic model diverges from traditional blockchain architectures by eliminating reliance on block rewards for network security. Instead, it introduces a market-driven security mechanism where the safety level of processes is directly correlated with users' needs and their corresponding willingness to pay for insurance. This shift aims to optimize resource utilization and align incentives across the network, fostering economic efficiency and reinforcing long-term system resilience.

To conclude, the AO protocol proposes a new perspective on decentralized computing; one that prioritizes system malleability, custom-made user security, and economic efficiency. These design choices address not only common limitations of current iterations of widespread systems, but also expand the spectrum of applications and use cases that are possible with a combined, trustless computer that is shared by all that wish to participate in the network.

References

[1] *A Rollup-Centric Ethereum Roadmap.* `https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698`. Accessed: 2024-04-24.

[2] *Adding Turing Complete computation to a blockchain.* `https://bitcointalk.org/index.php?topic=428589`. Accessed: 2024-04-24.

[3] *Adobe Photoshop in the Browser Thanks to Emscripten, Web Components and Project Fugu.* `https://www.bram.us/2021/10/27/adobe-photoshop-in-the-browser-thanks-to-emscripten-web-components-and-project-fugu/`. Accessed: 2024-04-24.

[4] *Akash Network Whitepaper.* `https://whitepaper.io/document/633/akash-network-whitepaper`. Accessed: 2024-04-24.

[5] Mustafa Al-Bassam et al. *LazyLedger: A Distributed Data Availability Ledger With Client-Side Smart Contracts.* 2019. URL: `https://arxiv.org/pdf/1905.09274.pdf` (visited on 04/24/2024).

[6] *ANS-104: Bundled Data v2.0 - Binary Serialization.* URL: `https://github.com/ArweaveTeam/arweave-standards/blob/master/ans/ANS-104.md`.

[7] ar.io. *Arweave Block Header 1278126.* 2023. URL: `https://arweave.net/block/height/1278126`.

[8] Arweave. *Arweave.* 2024. URL: `https://arweave.org` (visited on 04/24/2024).

[9] *Candle Whisper - Speech Recognition in WASM.* `https://huggingface.co/spaces/lmz/candle-whisper`. Accessed: 2024-04-24.

[10] *Candle-phi1-phi2 Wasm Demo.* `https://huggingface.co/spaces/radames/Candle-phi1-phi2-wasm-demo`. Accessed: 2024-04-24.

[11] Celestia. *Celestia: The First Modular Blockchain Network.* 2024. URL: `https://celestia.org` (visited on 04/24/2024).

[12] Bogdan S Chlebus, Dariusz R Kowalski, and Mariusz A Rokicki. "Average-time complexity of gossiping in radio networks". In: *International Colloquium on Structural Information and Communication Complexity.* Springer. 2006, pp. 253–267.

[13] Wikipedia contributors. *Actor model.* 2024. URL: `https://en.wikipedia.org/wiki/Actor_model` (visited on 04/24/2024).

[14] Wikipedia contributors. *Single system image.* 2024. URL: `https://en.wikipedia.org/wiki/Single_system_image` (visited on 04/24/2024).

[15] *Ericsson to WhatsApp: The Story of Erlang.* `https://thechipletter.substack.com/p/ericsson-to-whatsapp-the-story-of`. Accessed: 2024-04-24.

[16] Erlang.org. *Erlang Programming Language.* 2024. URL: `https://erlang.org` (visited on 04/24/2024).

[17] Forward Research. *Introducing the Universal Data License*. Accessed: 2024-04-24. 2024. URL: `https://mirror.xyz/0x64eA438bd2784F2C52a9095Ec0F6158f847182d9/AjNBmiD4A4Sw-ouV9YtCO6RCq0uXXcGwVJMB5cdfbhE`.

[18] Ethereum Foundation. *World Computer*. `https://www.youtube.com/watch?v=j23HnORQXvs`. Accessed: 2024-05-24. 2016.

[19] Carl Hewitt. *A Universal Modular Actor Formalism for Artificial Intelligence*. 1973. URL: `https://www.ijcai.org/Proceedings/73/Papers/027B.pdf` (visited on 04/24/2024).

[20] IETF. *RFC 768: User Datagram Protocol*. Accessed: 2024-04-24. 1980. URL: `https://www.ietf.org/rfc/rfc768.txt`.

[21] IETF. *RFC 791: Internet Protocol*. Accessed: 2024-04-24. 1981. URL: `https://datatracker.ietf.org/doc/html/rfc791`.

[22] Butler W Lampson, Nancy A Lynch, and Jørgen F Søgaard-Andersen. "Correctness of at-most-once message delivery protocols". In: *Proceedings of the IFIP TC6/WG6. 1 Sixth International Conference on Formal Description Techniques, VI*. 1993, pp. 385–400.

[23] Forward Research. *ao Token and Subledger Specification*. `https://cookbook_ao.g8way.io/references/token.html`. Accessed: 2024-05-24. 2024.

[24] *Scaling Ethereum with Layer 2 Rollups*. `https://l2beat.com/scaling/summary`. Accessed: 2024-04-24.

[25] Arweave Team. *SmartWeave*. `https://github.com/ArweaveTeam/SmartWeave`. June 2022.

[26] *Urbit Whitepaper*. `https://media.urbit.org/whitepaper.pdf`. Accessed: 2024-04-24.