Algorand Consensus Incentivisation

AN ALGORAND FOUNDATION DISCUSSION PAPER

John Woods
john@algorand.foundation
john@postquantum.dev

Michele Treccani
michele@algorand.foundation

John Jannotti
jj@algorand.com

Naveed Ihsanullah
naveed@algorand.foundation
naveed@jamsni.com

Version 1.0, 14th December 2023

1 Purpose

The goal of this project is to engineer a native solution at layer 1, modifying the Algorand protocol to *incentivise* the execution of consensus via block production rewards in Algo, Algorand's native token.

The technical goals of this project are:

1. Define the economic aspects of the design (source of funds, payout function, non-functional requirements).
2. Define the technical design (protocol level augmentations, code-level changes, functional requirements).
3. Define the approach to mitigate gamification ("Key surrender", "Absenteeism", "Vote neglect").

Contents

1 Purpose **1**

2 Acknowledgements **3**

3 Context **4**

4 Solution design **5**
4.1 Incentives from Fees 5
4.2 Economic considerations 5
4.3 Payout function 6
4.4 Frequency 7
4.5 Game mitigations/Anti-fragility 8
4.5.1 Absenteeism 8
4.5.2 Pooling 10
4.5.3 Modified Clients 11
4.6 Fees 12

5 Conclusion **13**

2 Acknowledgements

The ideas presented herein are inspired by and based on discussions with individuals from Algorand Foundation & Algorand Technologies (previously Inc), including the research and engineering groups, specifically: Silvio Micali, Paul Riegle, Gary Malouf, Eric Wragge, & Bruno Martins.

3 Context

Algorand employs Pure Proof of Stake (glossary below) as a means of reaching consensus. The security of the decentralised network is predicated on the total count of Algo tokens which are actively staked at any given moment, as a percentage of the total circulating supply. Notably, Algorand's liveness relies upon at least ∼80% of the participating ("online") stake to actively and honestly take part in the consensus protocol in order to produce blocks.

Algorand's original thesis contended, a priori, that a sufficient number of end-users would naturally be incentivised to run a node and contribute to online stake, motivated by the desire to secure their own funds, and as a corollary the network. However, a posteriori, it has been observed that as a consequence of the cost to run a participation node with high uptime, and indeed the technical expertise required to both instantiate and maintain a node, a critical mass of end-user contribution has not been reached.

In fact, as the distribution of Algo across accounts has become more uniform, and a larger proportion of the supply has become held by an increasingly less concentrated set of individuals, online stake has notably *dropped*. The aforementioned friction is seemingly too much for most individuals to contribute actively to network security.

This paper proposes a potential solution, at a protocol level, to encourage end-users to actively participate with their Algo. Additionally, we introduce the concept of incentives to reward participation, with the aim of materially increasing the total percentage of Algo in circulating supply which is participating in consensus.

4 Solution design

Here we outline the technical solution design for payment of consensus incentives, dynamically by the protocol, without manual intervention or processes.

4.1 Incentives from Fees

Henceforth, we use the term "mined incentives" to refer to the proportion of incentives yielded from the act of proposing a block which are directly attributable to Fees. As of 2023, the global Fee volume is unsubstantive. That said, we should introduce this mechanism now as any sustainable approach to consensus incentivisation will necessitate the consumption of Fees to pay block proposers.

The mining mechanism must be built into the L1 protocol, as it diverts funds from the global Fee Sink. The following technical approach should be sufficient to correctly pay out mined incentives.

1. When `proto.EnableMining`, add `Proposer` and `FeesCollected` fields to the block header.
2. Determine `proto.MiningPercent` which is the percentage of Algo paid in Fees that should go to the proposer of the block instead of the Fee Sink.
3. At the start of block `n+1`, move `MiningPercent * block[n].FeesCollected` from the Fee Sink to `block[n].Proposer`.
4. Outlaw spending from the Fee Sink to ensure the funds above are available in block `n+1`.

The movement is executed at the start of a given block `n+1` instead of during block `n` because a multi-participant node builds a block *before* it knows which of its accounts might actually be the proposer. Between blocks, the funds remain in the Fee Sink.

4.2 Economic considerations

In addition to the incentives generated by the Fee market, in the short to medium term additional funds will be required to ensure incentives are substantive whilst the network's transaction volume increases. Thus, the Algorand Foundation will contribute an additional sum of Algo to supplement the incentives generated by Fees.

To the end user, the Block-Reward paid to the proposers account will be congruent; that is, the portion of the reward which represents the AF contribution, and the portion of the reward which represents the Fee contribution will seamlessly combine.

4.3 Payout function

The supplemental incentives described above will payout following a time-dependent function, where the emission implies deflation. When supplemental incentives are exhausted, Fees scaled by `proto.MiningPercent` will become the sole payout per block. Here, we explore potential payout functions.

Consider the following two emission function definitions:

Defined below is a linear reward function, which at a given point in time (defined by round number η) evaluates to a Block-Reward in Algo:

First we define the emission shape, with a linear decrease:

$$R(\eta) = (1 - \frac{\eta}{\mathcal{N}})$$

then, adding a normalisation factor:

$$\mathcal{K} = \sum_{\eta=1}^{\mathcal{N}} R(\eta) = (\mathcal{N} - \frac{\mathcal{N}(\mathcal{N}+1)}{2 \times \mathcal{N}})$$

$\ni$ **the payout function is defined as:**

$$\mathcal{R}(\eta) = \mathcal{M} \times R(\eta) \times \frac{1}{\mathcal{K}}$$

Where:

$\mathcal{M}$: Total units for incentive (e.g., 200 million Algo).
$\mathcal{B}$: Block-time (e.g., 3 seconds).
$\mathcal{T}$: Total duration for the payout, expressed in seconds (e.g., 3 years).
$\mathcal{N}$: Total number of blocks over the payout period $= \frac{\mathcal{T}}{\mathcal{B}}$.

Whilst a linear payout is *reasonable*, we could also consider a function with a configurable exponential decay (also defined by round number η), which may be more fitting given Algorand's existing capped, inherently deflationary supply:

First we define the emission shape, with an exponential decrease:

$$R'(\eta) = e^{-\rho \frac{\eta}{\mathcal{N}}}$$

then, adding a normalisation factor:

$$\mathcal{K} = \sum_{\eta=1}^{\mathcal{N}} R'(\eta) = \frac{1 - e^{-\rho}}{1 - e^{-\frac{\rho}{\mathcal{N}}}}$$

$\ni$ **the payout function is defined as:**

$$\mathcal{R}'(\eta) = \mathcal{M} \times e^{-\rho \frac{\eta}{\mathcal{N}}} \frac{1}{\mathcal{K}}$$

Where:

ρ : Rate of decay.

Here, the shape of the curve would depend on the value of ρ: the lower the value of ρ the more elongated the curve.

Note: With the development of "Dynamic λ" (essentially dynamic round-times, where the time-to-wait for the lowest VRF proposal is dynamically calculated, and optimised to ignore some slowest percentile of proposers), it is likely prudent to consider a mechanism to temporally tweak the payout functions proposed above, as with dynamic round-times, η will no longer provide a linear approximation of time. As such, we may consider assuming an aggregate block-time for $\mathcal{B}$. Such an assumption can then be adjusted if significant deviation from our expected $\mathcal{T}$ occurs.

4.4 Frequency

The payout function for the Block-Rewards will execute on a periodic basis. The payout will occur every η block(s). At the time of writing the average block time is *3.3 seconds*, and the expected payout is *per round/block*.

4.5 Game mitigations/Anti-fragility

Below we describe the primary attack vectors and game-theory aspects as they pertain to the proposed implementation above.

4.5.1 Absenteeism

Concern:
Absenteeism refers to stakeholders registering their Algorand account online, but subsequently failing to participate in protocol operations (such as proposing and validating blocks). If a sufficient percentage of active stake declares itself online but does not do the work to participate in the consensus protocol the blockchain will stall. Recovery from this stall requires that the derelict stake eventually participates or a network hard-fork is executed to bypass the defaulting Algo from being considered.

Mitigation:
We propose a novel solution to Absenteeism which essentially empowers the network to suspend delinquent participants/ accounts from the staking set. Where that participant/account has failed to execute their incumbent responsibilities. As part of this solution we will introduce a new transaction type which will supplement the standard Key Registration Transaction transaction mechanism with a new type dedicated to "re-enrollment" of an account in the staking set should it have been suspended from said staking set by the network mistakenly.

This will ensure false positives have a mechanism by which they can re-engage in consensus without downtime. *Note: this solution operates at a consensus level.*

The approach is as follows:

- Consider one's stake always represents some $\mathcal{X}$% of $\mathcal{Y}$, one's proportion of the global stake.
- One's stake will imply a probability distribution across the staking set, defining a threshold with regard to the probable number of successful block proposals per account, a quantitative threshold which can be used to assess the likelihood of an individual acting in the best interest of the network.
- If an account falls foul of this threshold, the network will suspend the participant from the staking set, for example where an account's count of successful proposals is significantly below the statistically implied threshold.
- There is of course a possibility that an individual was just 'unlucky' rather than inactive. In such circumstances their node can trivially post a re-enrollment transaction which is, critically, signed by the node's Participation Key, not their Spending Key.
- Thus, the network can suspend misbehaving entities, including those who may not be bad actors but whose stake is unreliable and detrimental to the network, whilst simultaneously providing a mechanism for entities who were incorrectly suspended to reassert their commitment to the network and continue to earn rewards.
- To avoid suspension in cases where an account may simply receive a large portion of Algo which skews their implied threshold, a *constant* grace period will be observed prior to suspension to allow accounts to proactively respond, prior to any action being applied.
- This approach will utilise Participation Keys, and not Spending Keys. The process will also not require additional Key Registration Transaction transactions.

- This approach effectively provides "Garbage Collection" for consensus, permitting relatively significant portions of global stake (5% to 10%), within a reasonable *rolling* time period, to ungracefully exit consensus ad-infinitum without detriment to the network.
- As an additional check of liveness, nodes could be requested to ack in response to challenges based on Blockseeds, (which are notably unpredictable and impossible to influence). This approach would permit a targeted percentage subset of the active staking set to be "assessed", asserting their liveness, and suspended where an ack is not provided. This mechanism could be executed every $\mathcal{Z}$ rounds, with the set of participants based on some deterministic slice of the Blockseed, for example by comparing each staker's address with some $\mathcal{B}$ byte portion of the Blockseed. The challenge would also be predicated on Blockseed, and be trivially computable.
- In order to discourage unreliable consensus execution, when an account is suspended it should incur a small but substantive expense in order to return to online status. We may want to bound this cost with an integer limit value in the initial Key Registration Transaction, controlling the number of times an account can reassert. This bound allows a security conscious participant to be sure that their Participation Key cannot be used to "drain" their account.

Finally, minimum and maximum staking notionals may be enforced in order for an account to receive a Block-Reward.

A minimum, for example, $8,192$ Algo, $16,384$ Algo, $32,768$ Algo, would serve two purposes, first, it would set a hard upper bound on the total number of incentive-eligible participating nodes. This bound makes it practical to track all participating accounts in active memory, which is required to ensure absenteeism checks are time bound, second it would likely deter intermittent node operations by "nano" accounts.

A staking ceiling encourages large accounts to split their stake across multiple accounts, which, where split across multiple nodes, adds stall resilience to the network's consensus. Such a limit would likely be between 2^{26} and 2^{27} Algo.

4.5.2 Pooling

Concern:
The liveness of the Algorand protocol requires that ∼80% of the participating stake is held by honest participants. While participation in Algorand's consensus algorithm is neither computationally burdensome nor operationally complex in an absolute sense, it may still prove to be beyond the technical capability of the average Algorand user, who may not be familiar with managing computer systems. These participants may be likely to pool their Algo with a node operator who promises to participate on their behalf and return their rewards for an operational fee. Today this is possible without risking the Algo owner's principal because participation activity involves a secondary Participation Key, associated with, but not controlling spend from the payment account. If users choose to delegate their Participation Key to a pooling service, this may concentrate stake with a single entity, posing a risk to network safety through potential accidents or malicious actions, where, as outlined in the section above on Absenteeism, 4.5.1 a stall could occur.

This scenario cannot be meaningfully mitigated by limiting the maximum participation from a particular account, node or IP address as a Sybil-attack will trivially bypass such protections.

Mitigation:
Some high-level mitigations and anti-fragility measures have already been baked into the protocol. For example, if a Participation Key expires without re-registration, the network will gracefully remove the participant from the staking set. Additionally, the Foundation will work directly with potential operators to provide education and set operational best practice, including providing play-books.

Properly run pools may be beneficial to the network, where appropriate volumes of stake are separated over participation nodes which are professionally run and exhibit high-availability. Finally, an over-abundance of online accounts on a given participation node is naturally discouraged, running a participation node with more than ∼3 accounts present will result in a degradated performance during execution of consensus, notably resulting in lost rewards.

4.5.3 Modified Clients

Concern:
A strategy to earn consensus rewards whilst also minimizing the network and computational resources required whilst participating, is to participate through a bespoke, modified, Algorand node. The official Algorand node implementation maintains substantial local state reflecting the current balance tree of the Algorand blockchain. This reference implementation is usually executed on medium-end hardware to ensure that the responsibility of consensus execution (such as block proposal from current queued transactions and proposal verification based on current balances) can be completed in the smallest quanta of time. Network and computational resources may also be required for the message passing and algorithmic processing required. Thus, a sophisticated attacker may consider building a customised node implementation which avoids the primary resource intensive activities such as keeping in parity with other node's blockchains, proposing empty blocks or falsely validating proposed blocks without actually checking the payload - or equivalent shortcuts.

Mitigation:
Whilst we thought it prudent to consider the potential advantages from such an approach, there does not appear to be a meaningful benefit with this strategy. The computational burden from execution of the VRF and validation of contracts is negligible. The benefit derived from developing and executing such a customised node is not commensurate with the effort required to undertake such a strategy, in a Pure Proof of Stake context, it simply doesn't yield a material advantage.

4.6 Fees

As described above in 4.2, serious consideration must be given to the Fee dynamics in the short to medium term. Whilst we absolutely need to ensure Algorand stays inexpensive to use, we must also balance this with the need to secure the network in perpetuity. This may imply a Fee market, Fee increase or both. Notwithstanding, Algorand's vision remains unchanged. A vision which precludes a costly fee model.

5 Conclusion

We believe that incentivised consensus represents the right next step for the protocol. This design should yield a more decentralised and secure network, and critically, one which is secure in perpetuity, even in the context of a highly uniform token distribution.

The implementation of the changes discussed herein is a work in progress, the following PRs track that work:

- *PR 5740* - **Incentives: Implements "Mining" - diverting a portion of Fees to proposers**: github.com/algorand/go-algorand/pull/5740

- *PR 5757* - **Incentives: Suspend "absentee" accounts that don't propose**: github.com/algorand/go-algorand/pull/5757

- *PR 5799* - **Incentives: Heartbeat to keep an unlucky node online**: github.com/algorand/go-algorand/pull/5799

Barring unforeseen issues, we expect production (mainnet) roll-out of consensus incentives by mid 2024.

Glossary

Block-Reward a sum of Algo paid to the participation account of an active Staker, when the Staker has successfully produced a valid and network-wide accepted block.. 5–7, 9

Blockseed a pseudo-random number generated for each block. Its value is based on various pieces of information from the blockchain. It is used as part of the sortition algorithm.. 9

Fee Sink an account, (Y76M3MSY6DKBRHBL7C3NNDXGS5IIMQVQVUAB6MP4XEMMGVF2QWNPL226CA which receives all Fees generated by transactions on Algorand mainnet.. 5

Key Registration Transaction a native Algorand transaction type that is used to register an underlying account's balance as an active part of the global stake utilised by Consensus.. 8, 9

Participation Key a secondary private key, associated with the primary private key/account via signature from the primary private key, imbued with the right to sign messages on behalf of the associated account during consensus.. 8–10

Pure Proof of Stake a consensus algorithm based on verifiable random functions over elliptic curves, which provides a mechanism of cryptographic sortition. Naturally incentivises non-pooled block production.. 4, 11

Spending Key the primary private key for a given Algorand account, confers the right to spend/move the account's funds upon the holder.. 8

The information provided herein is for informational purposes only and should not be construed as financial, legal, or investment advice. It is the responsibility of any person who accesses the information herein to observe all applicable laws and regulations of their relevant jurisdiction. By proceeding to obtain the information, you are representing and warranting that all the applicable laws and regulations of your jurisdiction allow you to access such information. Algorand Foundation and our affiliates make no representations or warranties of any kind, express or implied, regarding the accuracy, completeness, or reliability of the information contained herein. Algorand Foundation and our affiliates assume no liability for any losses or damages that may result from reliance on the information contained in this paper.

This paper may contain forward-looking statements that are subject to risks and uncertainties that could cause actual results to differ materially from those expressed or implied by such statements. Forward-looking statements in this paper represent the judgment of Algorand Foundation and our affiliates as of the date of the paper. We do not undertake any obligation to update or revise any forward-looking statement to reflect new information or future events. You should not place undue reliance on forward-looking statements contained in this paper.

For the complete version of this disclaimer, please see our website: www.algorand.foundation/disclaimers