Algorand Blockchain Features Specification

Version 1.0

January 29, 2024

Contents

1 Blockchain and Balance Table

The Algorand blockchain is a sequence of blocks. Initially, it consists of a single block B_0, called the genesis block. New blocks are appended incrementally. Once block B_r is finalized, a consensus protocol is run to generate and finalize block B_{r+1}.

Each block B_r contains a set of transactions and a random seed Q_r, together with metadata that support cryptographic proofs of validity. Transactions are detailed in Section **??**.

The Algorand blockchain maintains a set of accounts owned by users. Each account is identified by an account public key (APK), a unique 32-byte string. The balance table S_r contains balance entries for each user, including:

- APK_u: account public key
- br_u: account balance
- User status (online/offline)
- Optional keys for participation in the consensus protocol

Minimum balance per account is 0.1 Algo to mitigate spamming, and the smallest unit is 1 microAlgo (10^{-6} Algo).

2 Key Management

Algorand employs the Ed25519 signature scheme. It supports multi-signature addresses for secure transactions. A multi-signature address is computed as:

$$APK = \text{Hash}(PK_1, PK_2, PK_3, \text{threshold} = 2) \tag{1}$$

where transactions require at least two valid signatures from the set of registered keys.

2.1 Verifiable Random Function (VRF) Keys

Users register a VRF public key in their balance entry. The VRF function is denoted as:

$$VRF(x) = \text{Signature of } x \text{ using VRF secret key} \tag{2}$$

The VRF ensures unbiased randomness for committee selection.

3 Consensus Protocol

3.1 Period (r, p) Voting Instructions

When user u starts period (r, p), they reset their timer to 0. The voting instructions are as follows:

1. **Step 0: Proposal** - When $\text{timer}_u = 0$:
   - If $p = 0$ or $p > 0$ and u has received a next-quorum for $\perp$ from period $(r, p-1)$, then u assembles a new block proposal B_u and propagates B_u and $H(B_u)$.
   - Otherwise, if $p > 0$ and u has received a next-quorum for $H(B') \neq \perp$ from $(r, p-1)$, then u propagates $H(B')$.
2. **Step 1: Filtering** - When $\text{timer}_u = 2\lambda$ (if $p > 0$) or $2\lambda_0$ (if $p = 0$):
   - If $p = 0$ or if $p > 0$ and u has received a next-quorum for $\perp$, then u selects the proposal with the minimum credential and soft-votes for it.
   - Otherwise, if $p > 0$ and u has received a next-quorum for $H(B') \neq \perp$, then u soft-votes for $H(B')$.
3. **Step 2: Certifying** - While $\text{timer}_u \in (2\lambda, \max(4\lambda, \Lambda))$:
   - If u receives a soft-quorum for $H(B)$ and a valid block B with $H(B) = H(B)$, then u cert-votes for $H(B)$.
4. **Steps 3-252: Recovery** - When $\text{timer}_u = \max(4\lambda, \Lambda) + 2^{s-3}\lambda + r$ ($r \in [0, 2^{s-3}\lambda]$):

- If u has seen a valid block B and a soft-quorum for $H(B)$, then u next-votes for $H(B)$.
- Otherwise, if $p > 0$ and u has received a next-quorum for $\perp$, then u next-votes for $\perp$.
- Otherwise, if u has received a next-quorum for $H(B') \neq \perp$, then u next-votes for $H(B')$.

5. **Steps 253-255: Fast Recovery** - When $\text{timer}_u = k\lambda_f + t$ ($t \in [0, \lambda_f]$):

   - If u has seen a valid block B and a soft-quorum for $H(B)$, then u late-votes for $H(B)$ (step 253).
   - Otherwise, if $p > 0$ and u has received a next-quorum for $\perp$, then u down-votes for $\perp$ (step 255).
   - Otherwise, if u has received a next-quorum for $H(B') \neq \perp$, then u redo-votes for $H(B')$ (step 254).