Exploring Algorand

Öz, B., Hoops, F., & Matthes, F. (2023). “Blockchain-based Systems Engineering”. Lecture Slides. TU Munich.

Chair of Software Engineering for Business Information Systems (sebis)
Department of Computer Science
School of Computation, Information and Technology (CIT)
Technical University of Munich (TUM)
wwwmatthes.in.tum.de

Outline

Introduction

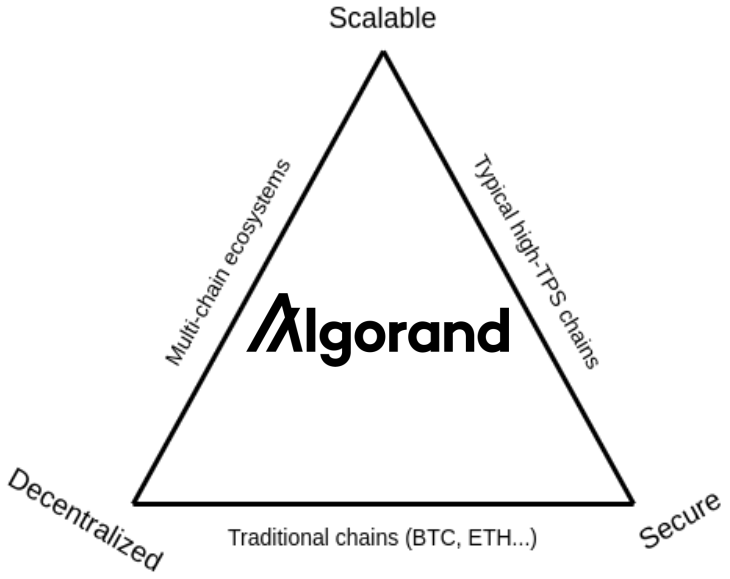
Algorand is a public, permissionless blockchain introduced by the Turing Award winner Prof. Silvio Micali (MIT) in 2017.

- Adopts a **Byzantine-Fault Tolerant** (BFT) consensus protocol combined with **Pure-Proof-of-Stake** (PPoS).
  - **No fixed set of consensus participants** or a **certain amount to be staked**.
  - Anyone can join consensus with any amount of stake.
  - **Voting power** in consensus is **proportional to the stake**.

Micali aims to **solve the blockchain trilemma** with Algorand.

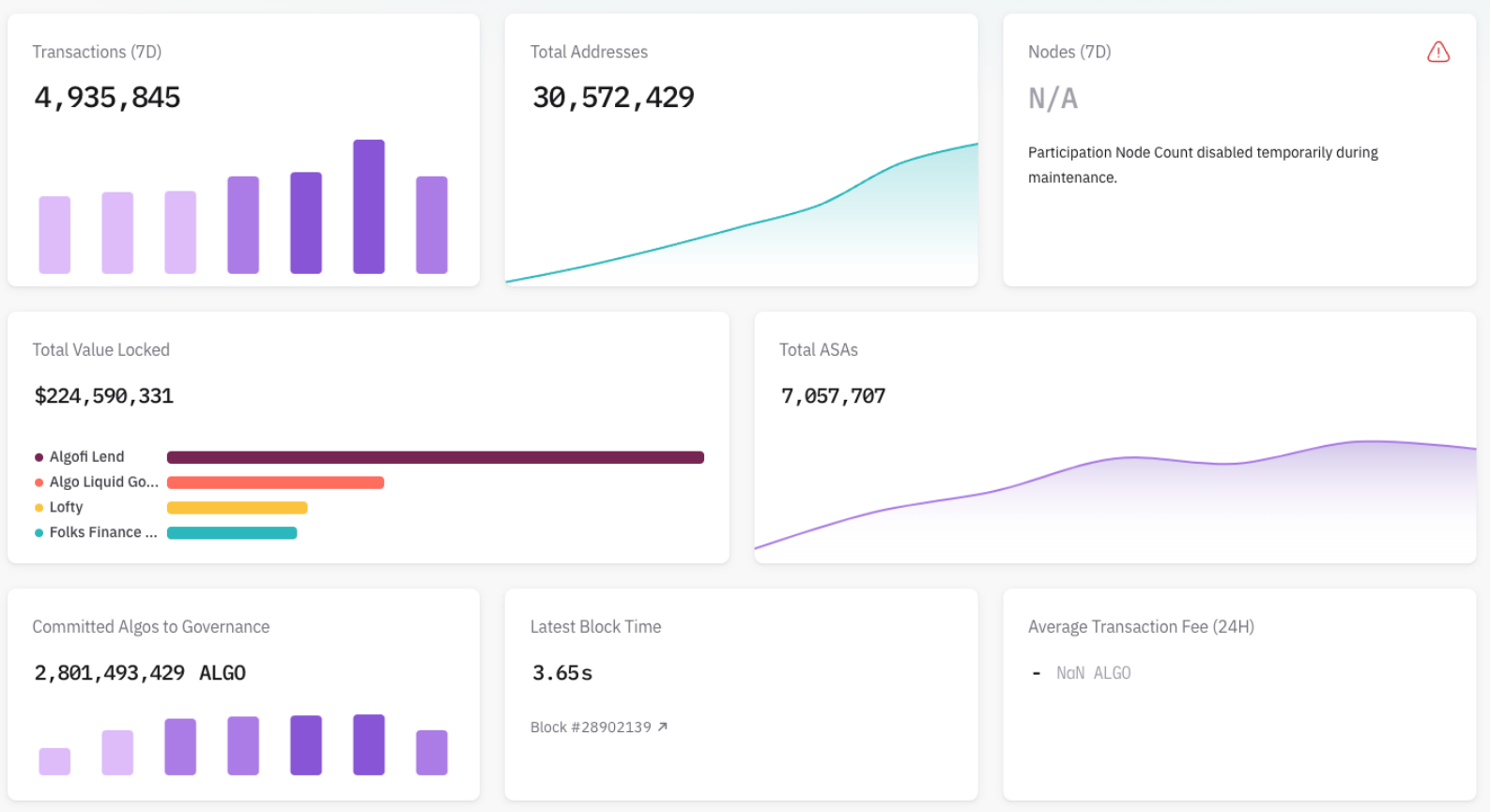
- ***Scalable***: Supports high transaction throughput and state growth
- ***Decentralized***: No single participant or group has a monopoly over how the chain runs
- ***Secure***: Robust against Byzantine (malicious) nodes up to a large percentage (ideally 50%)

Silvio Micali



The blockchain trilemma figure is taken from the following article by Vitalik Buterin: https://vitalik.ca/general/2021/04/07/sharding.html
Algorand whitepaper: https://algorandcom.cdn.prismic.io/algorandcom%2Fece77f38-75b3-44de-bc7f-805f0e53a8d9_theoretical.pdf

Network Metrics and Properties



Currency	**ALGO**
Block time	**< 3.9s**
Finality	**Immediate**
Block size	**5 MiB**
Max. Throughput	**6,000 transactions**
Transaction Fee	**0.001 ALGO[1]**
Circulating Supply	***7.2B***
Total Supply	***10B***

[1] 0.001ALGO ≈ $0.00017
ASA = Algorand Standard Assets (fungible + non-fungible)
Dashboard screenshot taken from: https://metrics.algorand.org/#/

11 Exploring Algorand - Öz, B., Hoops, F., & Matthes, F. (2023). “Blockchain-based Systems Engineering”. Lecture Slides. TU Munich.

Outline

1. Overview
2. Network, Consensus, and Incentives
3. Algorand Virtual Machine, Smart Contracts, and Standard Assets

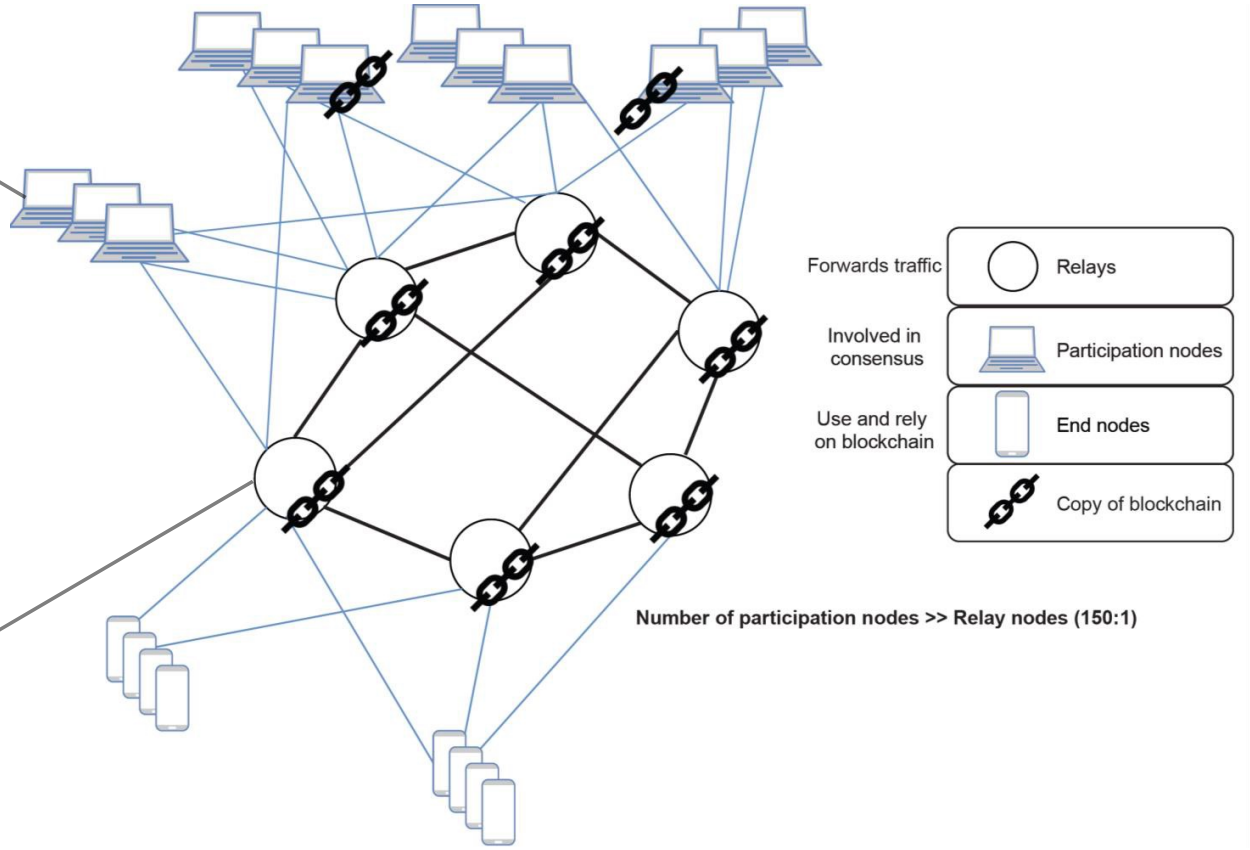
Network Architecture

Participation Node

- **Participates in the PPoS consensus**
- Holds ALGO stake (min 0.1 ALGO)
- **Connects only to the Relay Nodes**
- Can be run in *light* (stores only the last 1000 blocks) or *archival* (stores all the chain history) mode

Relay Node

- **Responsible for propagating transactions, blocks, and consensus messages**
- Connects to the other Relay Nodes and Participation Nodes
- **Routes communication to a set of connected Participation Nodes**
- Reduces communication hops due to highly efficient communication paths
- Currently, Relay Nodes are operated by Algorand Inc. and Algorand Foundation (**centralization**)



The Algorand network figure is taken from the slides of TUM Chair of Network Architectures and Services.
More info on Algorand node types: https://developer.algorand.org/docs/run-a-node/setup/types/

11 Exploring Algorand - Öz, B., Hoops, F., & Matthes, F. (2023). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

Consensus Overview

In Algorand, a consensus round consists of **three steps** until a **new block is appended** to the blockchain. A block is **finalized** if it has an associated **certificate**.

- A certificate includes valid **votes** and signatures from **committee members** participating in the block production round.

- Every account which holds ALGOs can attempt to participate in the consensus protocol. **The probability of getting selected to participate is proportional to the stake**.
- Repeats every ≈3.9 seconds.

Block Proposal

- **Randomly selected accounts propose** their **block** to the network
- Each proposed block is associated with a **hash value** which **determines** its **priority** (the lower, the better)
- ≈ 20 accounts are selected to propose a block each round

Soft Voting

- A randomly selected **committee votes** on block proposals to **filter down the proposed blocks to one**
- Vote on the block with the **lowest hash value**
- A quorum needs to be reached to move to the next step
- ≈ 2900 accounts are selected each round

Certify Voting

- Another randomly **selected committee** attempts to **certify the block which received a supermajority of votes** in the previous step
- If a quorum is not reached, the network will enter **recovery mode** (*safety* > *liveness*)
- ≈ 1500 accounts are selected each round

Selecting Consensus Participants

To randomly select the accounts to propose blocks or be committee members and vote, Algorand runs a **cryptographic sortition** algorithm. The algorithm determines a subset of accounts to participate in consensus based on the ALGOs held by each account.

The cryptographic sortition algorithm uses **Verifiable Random Functions** (VRF), which were introduced by Micali in 1999[1].

- Algorand's VRF takes as an **input** a **seed value** and **requires** a **secret/private key** to produce a **random hash value** and a **proof** that enables anyone who has the corresponding **public key** to **check the correctness of the hash value** with respect to the seed (just like verifying a signature with a public key).

- **Every account locally runs VRF for itself** and **checks whether they are eligible** to propose a block or become a committee member by **comparing the produced VRF hash to a threshold**[2]. A winning account propagates his VRF output to the network so others can verify it using his public key.

- To prevent Sybil attacks, the probability of winning depends on the ALGOs an account holds. Thus, VRF can be considered a **weighted lottery**.

Sends winning VRF proof to the rest of the network

$$VRF_{sk}(seed) \rightarrow \{hash, proof\}$$
$$if(hash < \mathrm{T}_{threshold}) \rightarrow Winner$$

$$Verify(seed, hash, proof, pk) \rightarrow \{true \,||\, false\}$$

[1] Verifiable Random Function: https://dash.harvard.edu/bitstream/handle/1/5028196/Vadhan_VerifRandomFunction.pdf

[2] This is a simplified and abstracted description how the sortition protocol works. A detailed explanation is available here: https://ihagopian.com/posts/the-intuition-behind-algorands-cryptographic-sortition

Selecting Consensus Participants (cont.)

- The **seed value** of a round is **determined by the last proposed block.** Hence, it is **unpredictable** and cannot be targeted by an adversary.

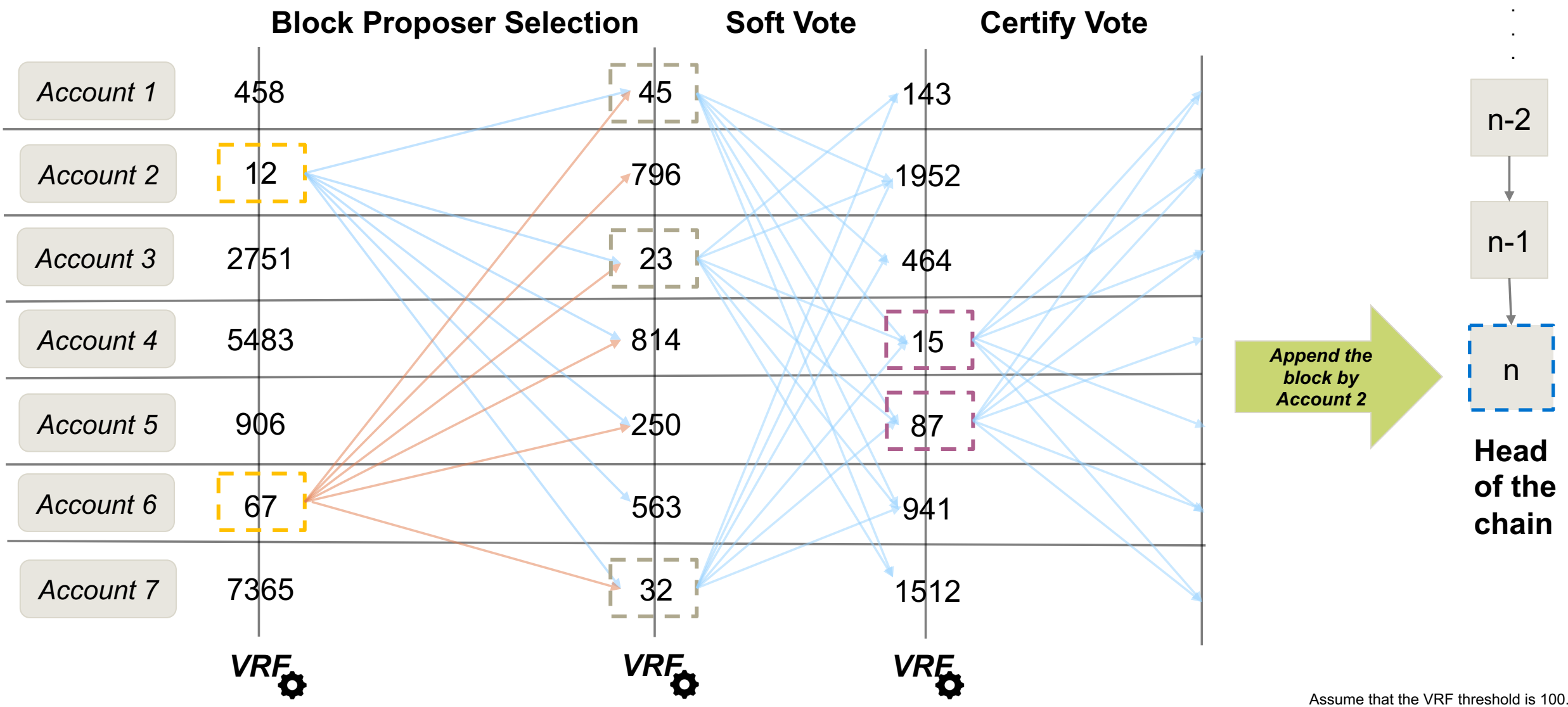
- **Until an account runs VRF for itself, no one can know whether that account is selected to partake in consensus, as VRF is dependent on the account's private key**. This way, it is prevented that an adversary targets an account involved in block production beforehand.

- Due to the probabilistic selection process, Algorand's PPoS consensus resembles the randomness of miner election in Proof-of-Work.

- Running the VRF protocol is **computationally cheap** (even a Raspberry Pi can do it).

- For security concerns, users **do not use the key pair they use when signing transactions for consensus as well**. Instead, they generate a new key pair (called a **participation key**) for a certain number of rounds and use that.

Simplified Consensus



Assume that the VRF threshold is 100.

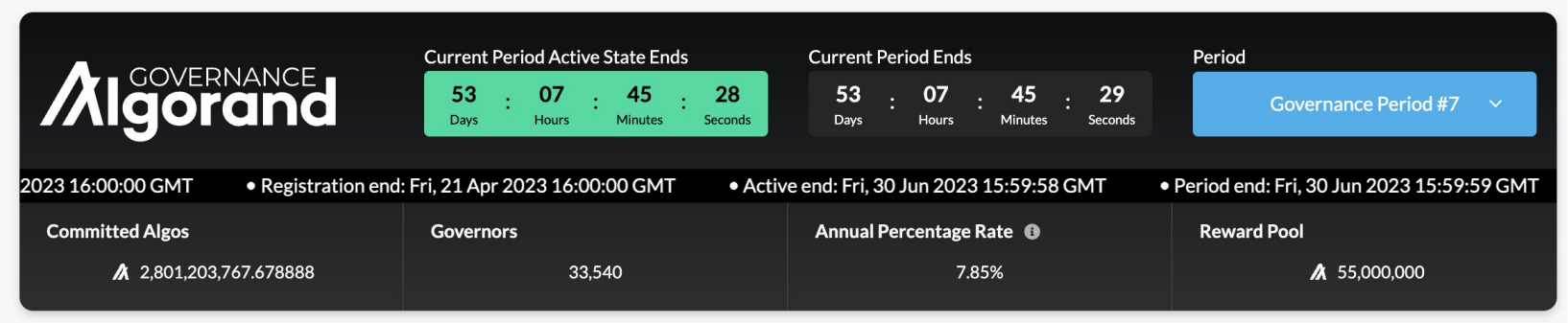
This slide is inspired from: https://www.youtube.com/watch?v=HWnewSr_B6k&ab_channel=AltExplainer

Incentives

In public, permissionless blockchains like Bitcoin and Ethereum, nodes are **incentivized to participate in consensus** through **block rewards** and **transaction fees**. In Algorand, the original reward scheme **rewarded all nodes** proportionally to their account balance. However, this scheme suffered from the **free-rider problem**.

- *Nodes lacked the incentive to spend energy on consensus as they are rewarded just by holding ALGOs.*

Since May 2022, participation rewards have been **replaced by governance rewards**[1]. Users can become governors and vote on proposals about ecosystem development. The weight of votes is proportional to the number of ALGOs that commit to governance for a three month period.



Currently, there is **no direct economic incentive for participation nodes to join consensus** as the protocol does **not issue block rewards,** and **transaction fees are collected in an Algorand-managed account**. There is active research going on in the community to find the optimal reward scheme for Algorand ([1], [2], [3]).

[1] More on governance: https://governance.algorand.foundation/governance-period-7
Governance screenshot taken from: https://algoexplorer.io/governance

Outline

1. Overview
2. Network, Consensus, and Incentives
3. Algorand Virtual Machine, Smart Contracts, and Standard Assets

Algorand Virtual Machine and Smart Contracts

Like the Ethereum Virtual Machine (EVM), Algorand has the **Algorand Virtual Machine** (AVM), a Turing-complete stack machine for **executing the state transition logic in a deterministic way**.

Algorand smart contracts, called **applications**, are stateful programs deployed on the blockchain, which anyone can remotely call to perform operations.

- A transaction interacting with a smart contract is called an application call transaction.
- **Every node runs AVM** to **evaluate application calls** and **determine** whether they **fail** or **succeed**.

TEAL - Transaction Execution Approval Language

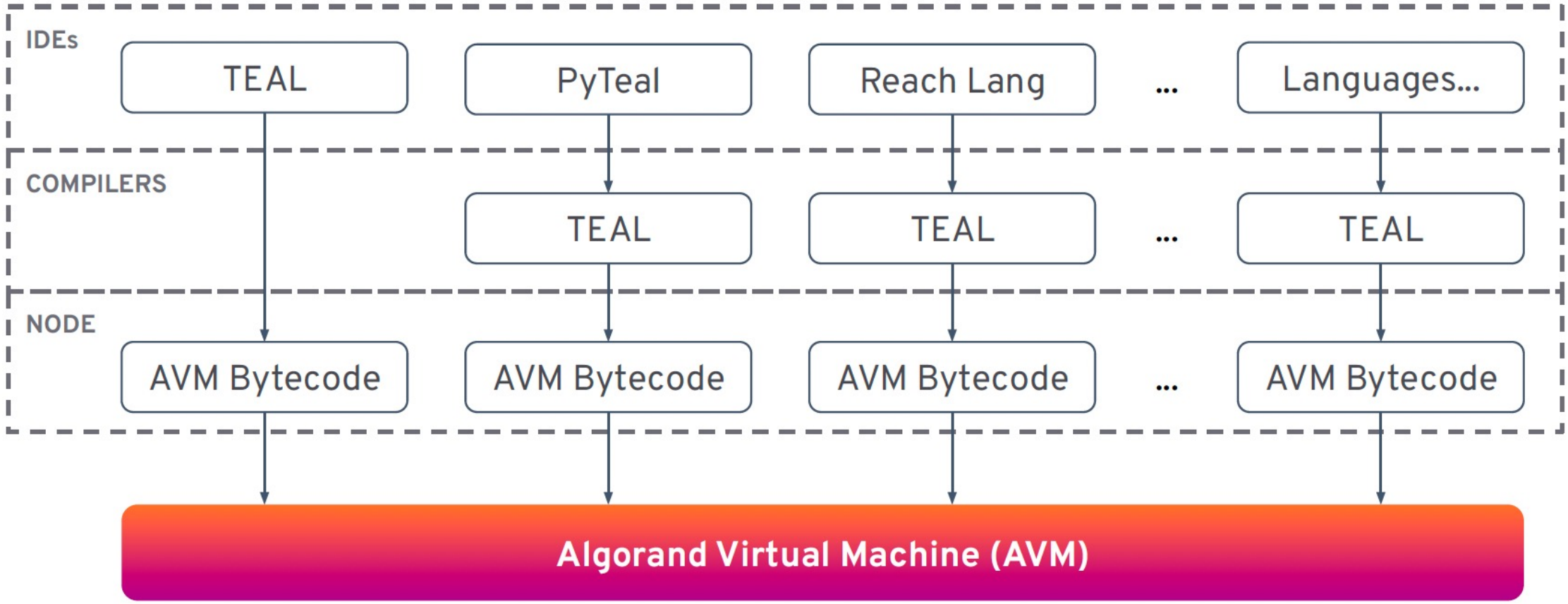
TEAL is the **assembly-like language** to **write smart contracts** on Algorand. It contains a set of opcodes that can be used to **implement application logic**.

- **Supports a wide range of operations**, including arithmetic, logical, and cryptographic operations, as well as control flow constructs like loops and branches.
- As TEAL is a low-level language, Algorand offers wrapper libraries like **PyTeal**, which offer a more familiar syntax.

Joe Polny from the Algorand Foundation will do a deep dive on AVM and TEAL in our guest exercise session on 12.07.2023.

More on AVM: https://developer.algorand.org/docs/get-details/dapps/avm/
TEAL: https://developer.algorand.org/docs/get-details/dapps/avm/teal/
PyTeal: https://pyteal.readthedocs.io/en/stable/

Algorand Virtual Machine and Smart Contracts (cont.)

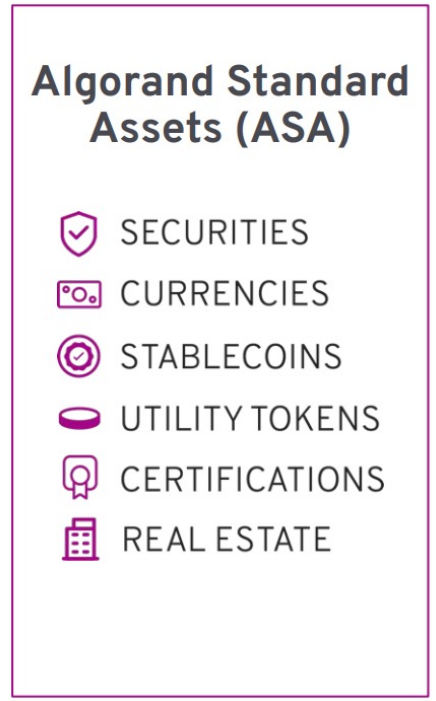


This illustration of AVM is taken from: https://github.com/cusma/algorand-school

Algorand Standard Assets

Algorand has its own set of asset standards, known as **Algorand Standard Assets** (ASA).

- ASA is a specification that **defines a common set of rules and functionalities for creating and managing digital assets** on the Algorand blockchain.
- It provides a simple and flexible way for users to create various types of assets.
- Similar to Ethereum's ERC token standards (ERC-20, ERC-721, ERC-1155).



Atomic swaps

Exchange assets without the need for a centralized exchange or intermediary.

Automatic asset freezing

Freeze assets in a specific account or group of accounts in case of a suspicious activity.

Clawback functionality

Revoke or claw back a specific amount of assets from a user's account.

Customizable metadata

Add metadata to assets, such as asset name, description, and image.

Smart contract integration

Compatible with decentralized applications and smart contracts on the Algorand network.

ASA figure is taken from: https://github.com/cusma/algorand-school
More on ASA: https://developer.algorand.org/docs/get-details/asa/

Algorand Standard Assets (cont.)

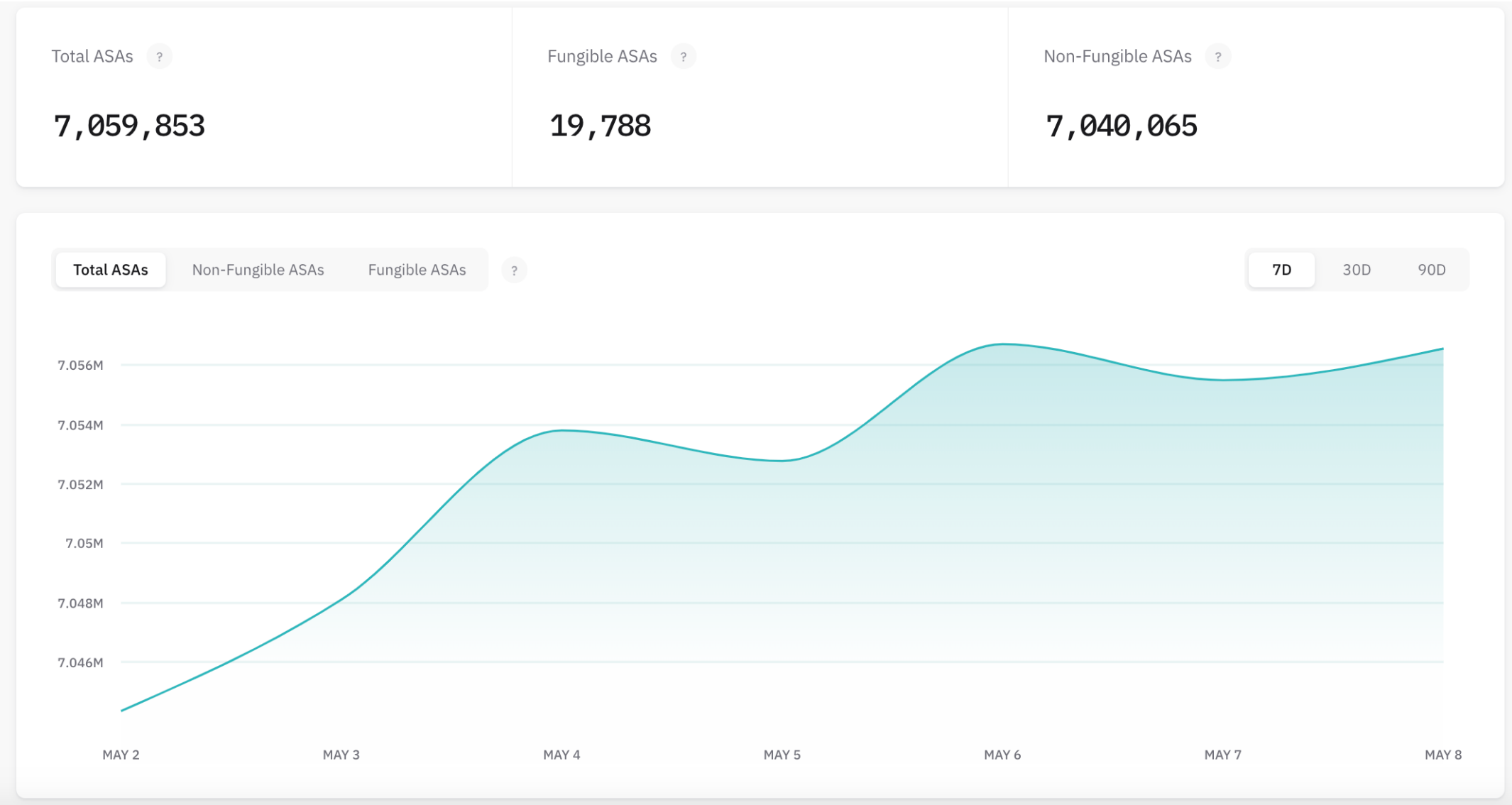


Figure taken from: https://metrics.algorand.org/#/protocol/#algorand-standard-assets

Algorand Standard Assets (cont.)

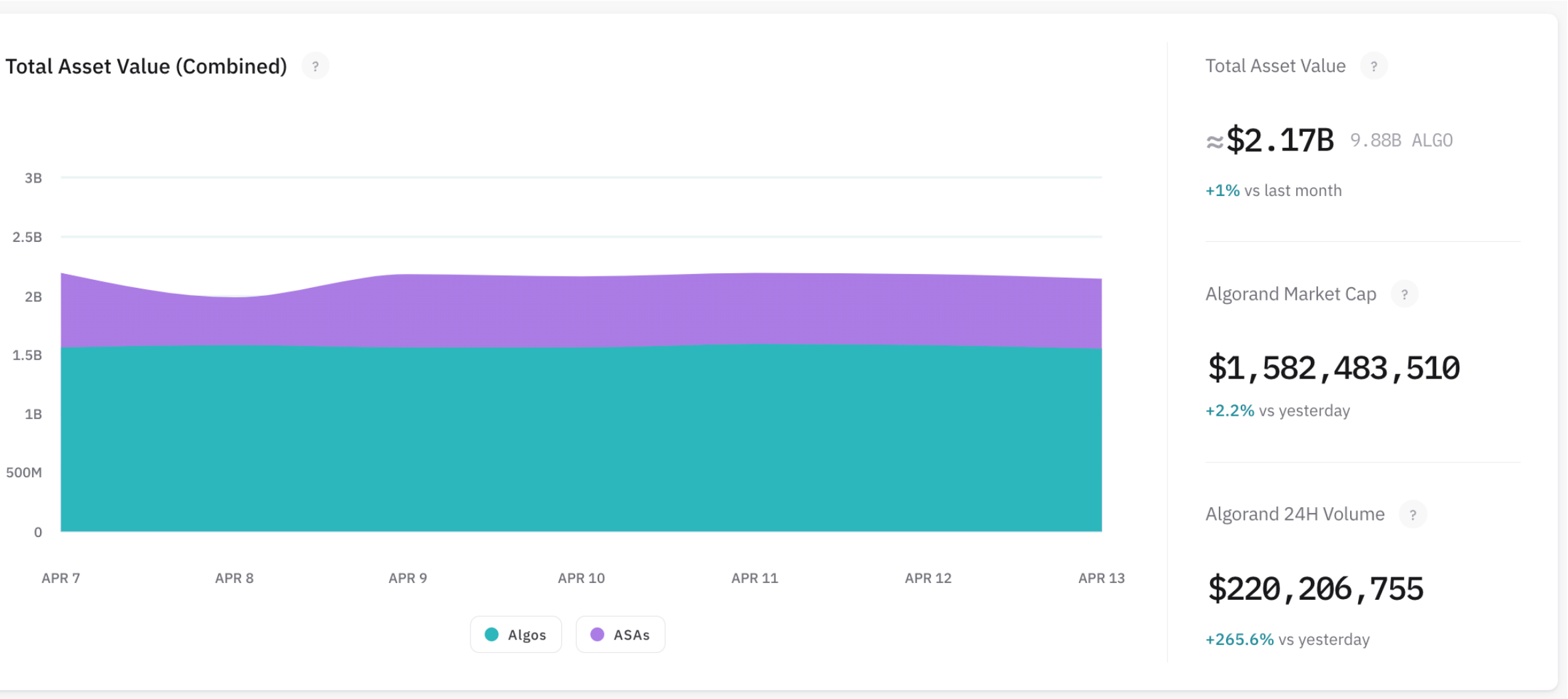


Figure taken from: https://metrics.algorand.org/#/value/

Useful Resources

- **Whitepapers**

https://algorand.com/technology/white-papers

- **Block Explorers**

https://algoexplorer.io/, https://algoscan.app/

- **Developer Docs**

https://developer.algorand.org/docs/

- **Consensus and Cryptographic Sortition**

https://developer.algorand.org/docs/get-details/algorand_consensus/

https://ihagopian.com/posts/the-intuition-behind-algorands-cryptographic-sortition

- **Metrics**

https://metrics.algorand.org/#/, https://app.metrika.co/algorand/dashboard/network-performance?tr=1d

- **Algorand vs. Ethereum**

https://developer.algorand.org/docs/get-details/ethereum_to_algorand/

- **Upgrades**

https://github.com/algorand/go-algorand/releases

- **More Resources**

https://github.com/aorumbayev/awesome-algorand, https://github.com/cusma/algorand-school/blob/main/algorand-school-english.pdf, https://www.algorand.foundation/general-faq