BASIC ALGORAND INFORMATION

Algorand Fundamentals

What is Algorand?

Algorand is a high-performance blockchain platform that aims to solve the blockchain trilemma by providing security, scalability, and decentralization simultaneously. It uses a Pure Proof-of-Stake (PPoS) consensus mechanism and was founded by Turing Award winner Silvio Micali.

Key Features

- Pure Proof-of-Stake (PPoS) consensus mechanism
- Near-instant transaction finality (<5 seconds)
- Low transaction fees (0.001 ALGO per transaction, ~$0.0002)
- Carbon-negative blockchain through sustainable practices and carbon credits
- Smart contract capabilities (using TEAL and PyTeal)
- Layer-1 solutions for tokenization, smart contracts, and atomic transfers
- Throughput of 1000+ TPS (transactions per second)
- Block time of approximately 3.9 seconds average

Native Asset: ALGO

- Primary utility token for the Algorand blockchain
- Used for transaction fees, staking, and governance
- Capped supply of 10 billion ALGO tokens
- Transparent distribution schedule

Network Types

- **Mainnet**: Production network for real transactions (https://algoexplorer.io)
- **Testnet**: Testing environment with test tokens (https://testnet.algoexplorer.io)
- **Testnet Faucet**: https://bank.testnet.algorand.network (dispenses test ALGO tokens)
- **Betanet**: Experimental features before testnet deployment
- **Sandbox**: Docker-based local development environment (https://github.com/algorand/sandbox)

Algorand Technical Architecture

Consensus Mechanism

- Pure Proof-of-Stake (PPoS) where validators are randomly selected based on stake
- Byzantine agreement protocol for reaching consensus
- Block finality achieved in a single round
- Cryptographic sortition for committee selection
- Byzantine fault tolerance protecting against attacks

Account Types

1. **Standard Accounts**: User-controlled accounts with public/private key pairs
2. **MultiSig Accounts**: Accounts requiring M-of-N threshold signatures
3. **LogicSig Accounts**: Contract accounts controlled by TEAL programs
4. **Application Accounts**: Smart contract accounts that control assets based on program logic

Account Details

- **Address format**: Base32-encoded 58-character string starting with "A"
- **Security**: Ed25519 cryptographic signatures
- **Minimum balance**: 0.1 ALGO required (increases with assets and applications)

- **Rekeying**: Ability to change private key while maintaining address and balances

Transaction Types

1. **Payment Transactions**: Transfer ALGO between accounts
2. **Asset Transactions**: Create, manage, or transfer ASAs (Algorand Standard Assets)
3. **Application Transactions**: Create or interact with smart contracts
4. **Key Registration Transactions**: Register participation keys for consensus
5. **Asset Freeze**: Freeze or unfreeze assets in specific accounts
6. **State Proof**: For interoperability and bridge security
7. **Atomic Transfers**: Group multiple transactions that either all succeed or all fail (up to 16 transactions)

Algorand Standard Assets (ASA)

- Native token standard on Algorand (similar to ERC-20 on Ethereum)
- Features: fractional ownership, role-based asset control, forced transfers
- Metadata support and asset clawback capabilities
- Low cost to create (~0.1 ALGO plus minimum balance requirement)
- Creator-defined parameters: Total supply, decimals, unit name, asset name, URL, metadata hash
- Role-based access: Manager, reserve, freeze, and clawback addresses
- Opt-in mechanism: Users must explicitly opt in to receive assets
- Fractional NFTs: Supported through asset decimals configuration

State Storage

- **Global State**: Accessible by all users of the application (limited to 64 key-value pairs)
- **Local State**: Per-user storage (limited to 16 key-value pairs per user)

- **Box Storage**: Additional variable-sized storage space beyond global/local state (8 KB minimum)

Algorand Development Environments

Algorand Sandbox

- Docker-based local development environment
- Includes complete Algorand node and indexer
- Configurable to match different network versions
- Useful for testing applications in isolated environment

Components

- Algod (consensus node)
- Indexer
- Postgres
- KMD (Key Management Daemon)

Launch

```
git clone <https://github.com/algorand/sandbox.git>
cd sandbox
./sandbox up
```

Integration Points & APIs

Indexer API

- Search and filter blockchain data
- Track transactions, assets, and applications
- Query historical state
- Monitor account activity

Node API

- Submit transactions
- Query current state
- Access blockchain status
- Manage accounts and keys

REST API Endpoints

- `/v2/transactions`: Transaction submission and retrieval
- `/v2/accounts`: Account information and assets
- `/v2/applications`: Smart contract state and information
- `/v2/assets`: Asset configuration and holdings

Standards and Specifications

Interoperability and Standards

- **ARC3**: NFT standard
- **ARC4**: Application binary interface
- **ARC19**: Metadata hosting standard
- **ARC69**: Alternative NFT metadata standard
- **ARC200**: Alternative fungible token interface
- **State proofs**: Chain verification for bridges
- **TEAL Templates**: Common smart contract patterns

Technical Specifications

- **ALGO supply**: 10 billion total, with transparent distribution schedule
- **Minimum balance requirements**: Account = 0.1 ALGO, Asset = 0.1 ALGO, App = 0.1 ALGO
- **Global state**: 64 key-value pairs maximum per application
- **Local state**: 16 key-value pairs maximum per user per application

- **Box storage**: Variable-sized storage with 8 KB minimum
- **Maximum group size**: 16 transactions per atomic transfer
- **Maximum application size**: 2KB approval program + 1KB clear program
- **Network upgrades**: Protocol version changes through on-chain governance
- **Block size**: Variable with average of 5,000 transactions per block

Educational Resources

- **Developer portal**: https://developer.algorand.org/learn
- **Algorand University**: https://algorand.foundation/university
- **Interactive tutorials**: https://learn.algorand.dev
- **Developer Discord**: https://discord.com/invite/algorand
- **Documentation**: https://developer.algorand.org/docs
- **GitHub repositories**: https://github.com/algorand
- **Grants program**: https://algorand.foundation/grants-program
- **Learning paths**: From beginner to advanced
- **Code examples and AVM walkthroughs**