Technical and Tokenomics Whitepaper

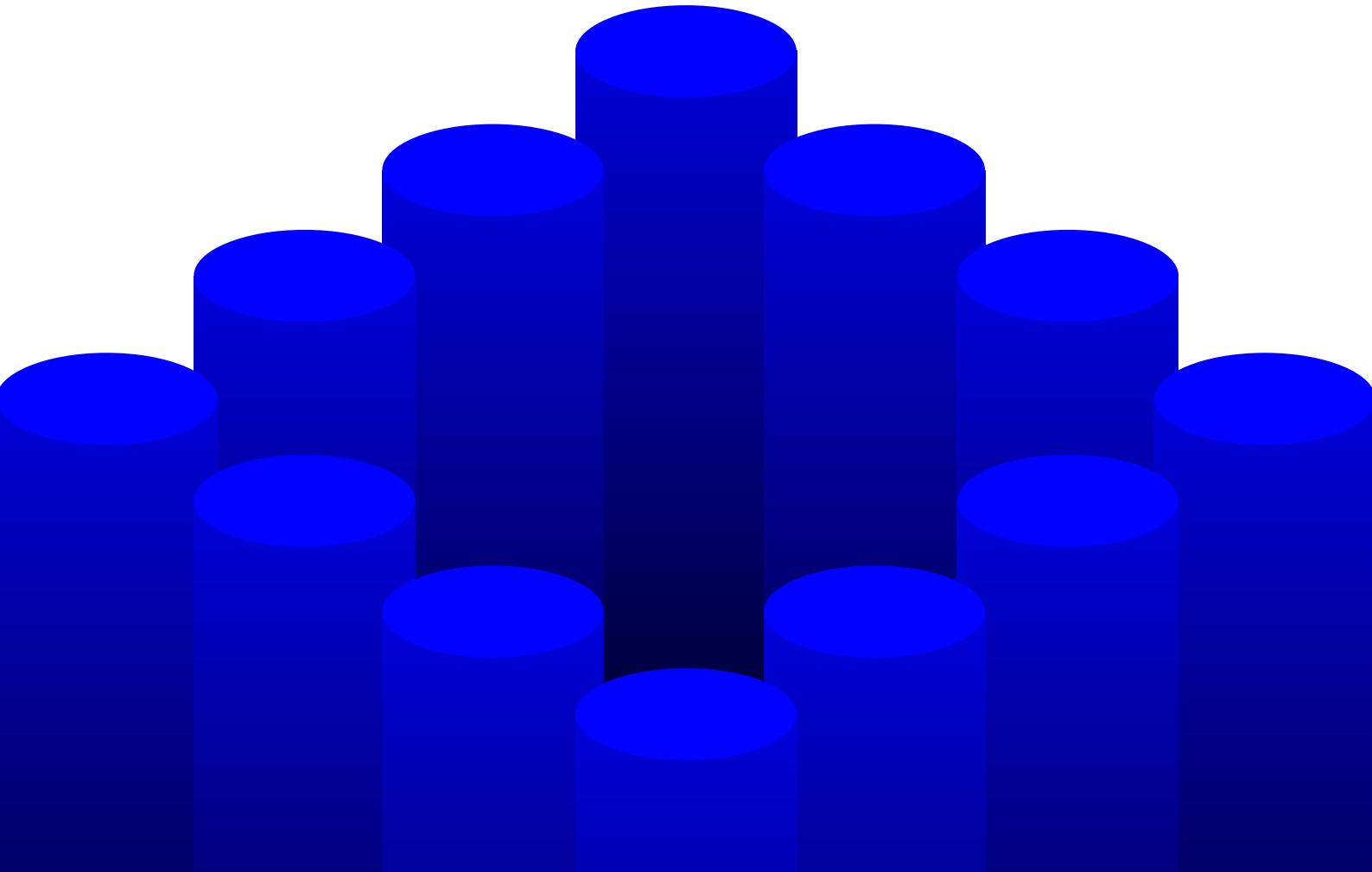


Table of Contents

1. Introduction

IOTA is an open-source distributed ledger technology designed to bring real-world applications on-chain. Scalable, efficient, and secure, it powers a public goods infrastructure optimized for enterprise solutions and Web3 innovation. With smart contracts and a modular architecture, the IOTA mainnet provides a global, permissionless foundation for diverse applications and a streamlined pathway to market adoption.

Backed by a proven track record in industry and public sector collaborations, IOTA has been applied to supply chain management, digital identity, energy systems, and more. It also empowers developers to build decentralized applications, DeFi projects, and custom blockchain-based solutions.

2. Tokenomics

The IOTA token ($IOTA) is the native asset and the foundation of economic activity within the IOTA network. Its design governs how users interact with the system, how resources are allocated, and how consensus is maintained. The tokenomics of IOTA are carefully designed to align with its vision, balancing incentives, network security, and sustainable growth.

With the rebase of the IOTA network, a total of 4,600,000,000 IOTA tokens were migrated from the previous Stardust network. All original tokens are now represented on the IOTA Mainnet as objects of type 0x2::iota::IOTA. These tokens remain accessible at the same hexadecimal addresses used on the previous Stardust-based network, and no manual migration is required. Each IOTA token is divisible into smaller units called NANOs, with one IOTA equivalent to one billion NANOs. A summary about circulating supply and inflation can be found at Appendix: Total and Circulating Supply.

$IOTA is also essential for securing the network through staking. Our protocol uses a delegated proof-of-stake (DPoS) mechanism, meaning that validator influence is determined by delegated stake. Users delegate their IOTA tokens to validators to support network security and earn a share of the minted subsidy. At the end of each 24-hour epoch, 767,000 new IOTA tokens are minted, corresponding to an initial annual minting rate of 6% of the total supply. The number of newly minted IOTA tokens per epoch is fixed, meaning that, in percentage, the minting rate changes over time. These tokens are distributed among participants based on validator performance, commission rates, and the amount of delegated stake.

The IOTA token has several key functions beyond being a medium of exchange. Firstly, it is used to pay for transaction fees and storage deposits on the network. Every transaction incurs a small fee that is automatically burned, reducing the overall token supply. Additionally, transactions that increase the storage burden on validators require a small locked deposit. This deposit is returned to the user once this storage demand is released.

Thus, the total supply of IOTA tokens fluctuates over time, influenced by two opposing forces: the minting of new tokens and the burning of fees. The subsidy minting is handled at the protocol level; while burning occurs automatically as a function of network usage. As a result, while the IOTA token supply is not capped, its growth is designed to remain at most linear over time.

In the sections that follow, we explore each component of the IOTA economy in more detail, including fees, staking mechanics, and validator dynamics.

2.1. Staking and Rewards

A set of independent validators participate on the IOTA network, each running its own instance

of the IOTA software on a dedicated machine or virtual server. Each validator handles read and write requests sent by clients, verifying transactions and updating on-chain information. Naturally, these activities are rewarded. In this section, we detail how this is done.

IOTA uses Delegated Proof-of-Stake (DPoS) to determine which validators operate the network and their voting power. The committee of validators that participate in consensus is formed, initially, by the 50 top stakers among candidates fulfilling minimum requirements (see 2.1.1. Validator Staking Pool Requirements). We aim to increase the maximum committee to 150 validators, enhancing decentralization without compromising performance. Those validators are incentivized to participate in good faith via staking rewards, which are slashed in case of misbehavior. We refer to a validator and its delegators by "staking pool".

IOTA adopts a self-custodial staking, i.e., stakers keep their staked IOTA tokens in their own account. IOTA token holders are then free to withdraw their IOTA or to change their selected validator at any time. To mitigate the security risks associated with centralization, a validator's voting power is capped at 10%. If a validator receives more than 10% of the total delegated stake, their effective voting power is limited to this threshold, with the excess stake redistributed among other validators. In the rest of this document, references to "voting power" or "pool stake" refer to these capped values.

The operation of the IOTA network is temporally partitioned into non-overlapping, approximate fixed-duration (~24-hour) epochs. During a particular epoch, the set of validators participating in the network and their voting power is fixed. At an epoch boundary, reconfiguration might occur and can change the set of validators participating in the network and their voting power. When a user stakes with a validator, their stake counts towards the validator's voting power starting from the next epoch. Similarly, when a user withdraws their stake, it stops counting from the next epoch. The distribution of staking rewards are also processed at epoch boundaries. Specifically, this process happens as follows:

1) At the beginning of each epoch E, the delegated tokens $poolStake(i,E)$ for a given validator i are added up, and a new validator committee is formed. Following this action, the protocol computes the total amount of stake $totalStake(E)$ as the sum of the delegated IOTAs among all the validators.

2) During the epoch, users submit transactions to the IOTA network and validators process them. For each transaction, users pay the associated computation fees, the storage deposit and, optionally, a tip for prioritization (see 2.2. Fee Structure). In cases where users delete previous objects or data within objects, users obtain a rebate of their storage deposit. Validators observe the behavior of other validators and evaluate each other's performance.

3) At the end of each epoch E, the protocol distributes staking rewards to the participants. This occurs through the following main steps:

   a) A maximum amount of stake rewards is assigned to each pool, proportionally to their stake and the total amount of stake rewards. The stake rewards are composed of a validator subsidy $subsidy$—currently set to 767,000 IOTA tokens per epoch—, plus the total amount of tips $tips(E)$ paid by users during the epoch. The maximum staking reward $maxPoolReward(i,E)$ for validator pool i at epoch E is computed as

   $$maxPoolReward(i,E) = \frac{poolStake(i,E)}{totalStake(E)} \times (subsidy + tips(E))$$

   b) The value above is then adjusted to consider validators who have not met minimum quality requirements. We denote as $performance(i,E)$ the performance factor of a validator i at epoch E, which corresponds to $performance(i,E) = 1$ for fully performant validators and $performance(i,E) < 1$ for non-performant validators. This variable ensures that validators have incurred monetary risk and therefore are incentivized to operate the IOTA network efficiently. The rewards actually distributed to each pool is computed as
   $poolReward(i,E) = performance(i,E) \times maxPoolReward(i,E)$

   c) Now, each pool's amount of rewards is distributed between the validator and the other parties. Validators first keep a commission $com(i,E)$ (of maximum 20%) of the total pool rewards as a fee to cover their costs. The rest of the rewards (i.e., after discounting the validator's commission) is distributed among all users proportionally to their stake. Specifically, let $stake(j,E)$ represent the stake owned by a delegator j, and delegated to pool i at epoch E. The rewards for delegator j and for the validator i itself in this epoch equal

   $$delegatorReward(j,E) = \frac{stake(j,E)}{poolStake(i,E)} \times (1 - com(i,E)) \times poolReward(i,E)$$

   $$validatorReward(i,E) = com(i,E) \times poolReward(i,E)$$

Consequently, validators can earn more the better their performance and the more delegated stake they can attract. On the other hand, IOTA token stakers will also receive less rewards when their selected validator is penalized due to poor performance. An unresponsive validator is thus doubly exposed to IOTA incentives: they lose directly through slashed rewards and indirectly through reduced user stake in future epochs as stakers move their tokens to more responsive validators.

2.1.1. Validator Staking Pool Requirements

There are minimum staking requirements a validator must satisfy to be eligible to be in the validator committee. More precisely:

- A validator candidate must accrue at least 2M IOTA of delegated stake before they can request to join the validator set.
- If an active validator's delegated stake falls below 1.5M IOTA, they have seven epochs of grace period to gain back the stake before being removed from the validator set.
- If an active validator's delegated stake falls below 1M IOTA, they are removed from the validator set at the end of the current epoch boundary.

2.2. Fee Structure

As an inherent part of the IOTA economic model and incentive structure, the IOTA transaction pricing mechanism achieves two outcomes:

1. Delivering low, predictable transaction fees.
2. Preventing spam attacks.

This enables users to focus on using the IOTA network to provide the best user experience without needing to forecast the current market price of gas fees. Since validators receive subsidies for doing their job correctly, they are not solely reliant on user fees to incentivize them.

Transaction fees on the IOTA network are intentionally minimal, designed to foster accessibility while maintaining economic sustainability. A portion of these fees is burned, introducing a deflationary mechanism which counterbalances the inflationary issuance of tokens used for staking rewards. Another portion of the fees is allocated to cover the cost of data storage on the ledger, tying the network's economic incentives to responsible resource usage. When data stored on the ledger is later freed, the tokens allocated for that storage are fully rebated, incentivizing efficient use of storage resources and ensuring the ledger remains optimized over time.

The net fees that a user pays equals the computation and storage fees minus the rebates associated with data deletion:

$net\_gas\_fees = computation\_fee + storage\_deposit - storage\_rebate$.

The information on net gas fees is displayed in the IOTA network explorer for each transaction block.

While computation fees and storage deposits are considered different payments, they are conceptually similar in that they each translate computation or storage into IOTA terms by multiplying computation or storage units by the relevant price. Since the storage component is 100% rebatable, the terms storage fee and storage deposit are used interchangeably.

Storage fees

Storage fees vary depending on the amount of new data written into on-chain storage. Each byte held in storage is mapped into 100 storage units. So, for example, a transaction whose effects imply the storage of 25 bytes will cost the price of 2,500 storage units. Importantly, the rebating mechanism incentivizes users to minimize the storage burden they place on all nodes by reducing their storage requirements and cleaning up unused objects. Finally, the storage fee for a transaction is given by

$$storage\_fee = storage\_units \times storage\_price,$$

where the $storage\_price$ is set by the protocol and the $storage\_units$ is known by the user in advance.

Computation fees

Different IOTA transactions require varying amounts of computational efforts for processing and execution. IOTA translates these varying operational loads into transaction fees by measuring each transaction in terms of computation units. In general, more complex transactions require more computation units.

IOTA charges gas consumption by rounding up computation units to the nearest 1,000, meaning that all transactions consume at least 1,000 computation units. The largest possible consumption is 5,000,000 computation units; if a transaction requires more than that, it aborts. The IOTA Gas Pricing is such that the computation component is given as follows:

$$computation\_fee = computation\_units \times gas\_price$$

where the user specifies the gas_price. This value must be greater than or equal to the $reference\_gas\_price$ defined by the protocol. The burned portion of these fees is given by

$$computation\_units \times reference\_gas\_price,$$

and any additional value not burned is considered a tip to validators and distributed together with rewards. Specifically, the tip value of

$$tip = (gas\_price - reference\_gas\_price) \times computation\_units \geq 0$$

is not burned and, instead, is distributed among participants (see 2.1. Staking and Rewards; note that the rewards distributed at epoch E are proportional to $subsidy + tips(E)$). However, during regular network operations, users can expect to have their transactions promptly included by paying only the burnt portion of the fees (i.e., by setting $gas\_price = reference\_gas\_price$).

Additionally to gas_price, the user must define a gas_budget. This provides a cap to the

amount of gas fees paid, since it is not possible to perfectly forecast how much a transaction costs before the user submits it to the IOTA network. The gas budget for an IOTA transaction is defined in IOTA units, and transactions are successfully executed if:

$gas\_budget \geq \{computation\_fees, net\_gas\_fees\}$.

Ultimately, a successful transaction requires the end user to pay the transaction's *net_gas_fees*. However, storage fees and rebates can only be calculated after the transaction is completely executed, meaning that, during execution, nodes can estimate lower bounds for *computation_fees* but not for *net_gas_fees*. The gas budget checks happen as follows:

- During execution, lower bounds for *computation_units* (and consequently, for *computation_fees*) are periodically calculated. The transaction will abort whenever its *gas_budget* cannot pay for these computation fees. In this case, the whole *gas_budget* is charged.
- In case the execution was not aborted (which means that *gas_budget* was large enough to cover *computation_fees*), the storage fees and rebates are calculated given the transaction effects.
  - If storage rebates are at least as large as the storage fees, *net_gas_feescomputation_fees*, meaning that the budget set by the user will certainly be high enough. In this case, the execution is successful and the user will be charged *net_gas_fees*.
  - If storage rebates are smaller than storage fees (meaning that *net_gas_fees* > *computation_fees*) and the budget set by the user is enough to pay for these *net_gas_fees*, the execution is successful and the user will be charged *net_gas_fees*.
  - Finally, if *net_gas_fees* > *computation_fees* but the budget set by the user not is enough to pay for these *net_gas_fees*, the transaction will fail. In this case, *computation_fees* are charged, as the storage fees associated with mutating the transaction's input objects.

Note that in some cases—especially in the presence of high storage rebates, and thus, negative net storage fees—the total gas fees the user pays (i.e., *net_gas_fees*) might be lower than the computation fees. Thus it is often the case that a transaction only goes through if the gas budget is considerably higher than the net gas fees that a user ultimately pays.

Importantly, the minimum gas budget is 1,000,000 NANOS or 0.001 IOTA (recall that any transaction uses at least 1,000 computation units and assuming that *reference_gas_price* is 1,000 NANOS).

3. Underlying Tech

IOTA is a next-generation distributed ledger technology designed to address the scalability, programmability, and decentralization limitations of earlier blockchain systems. Building on nearly a decade of research and development, IOTA integrates advanced innovations such as the Move Virtual Machine (MoveVM) to its architecture, creating a secure, scalable, and efficient platform for global financial and industrial applications.

IOTA employs a Byzantine Fault Tolerant consensus protocol (Mysticeti) to ensure robust network security and high throughput, achieving sub-second transaction finality and supporting tens of thousands of transactions per second. Mysticeti's underlying DAG structure eliminates traditional blockchain bottlenecks, ensuring the system remains efficient under high demand. Native digital asset functionality allows users to create and manage tokens directly within the protocol without reliance on smart contracts, reducing complexity and costs.

The IOTA ecosystem enables programmability in both Layer 1 and Layer 2 using Move and EVM/ Solidity smart contracts respectively:

Layer 1

Move is a powerful, secure programming language designed specifically for digital asset management and smart contracts. Its unique features make it an ideal choice for developers working in the blockchain space. Here are the key features of Move on IOTA:

- **Object-Centric Design:** Move is fundamentally object-centric, allowing developers to intuitively model complex data structures and interactions. In Move, you can define objects representing assets, users, contracts, and more, facilitating a natural and straightforward way to manage state and behavior.
- **Performance:** Move on IOTA is based on the object model, not a shared global state. This allows transactions to be executed in parallel, which translates into a network with high throughput, less congestion, and therefore lower gas fees.
- **Security:** Move prioritizes safety and efficiency, with Rust's ownership model inspiring Move's approach to memory management and resource control. The compiler enforces strict rules to prevent common programming errors, ensuring that assets remain secure within user accounts and cannot be accessed without the correct keys. While the compiler catches many potential issues before deployment, developers should still be mindful of logical errors, arithmetic overflows, and other runtime concerns when implementing smart contracts.

Layer 2

EVM stands for "Ethereum Virtual Machine" and is currently the tried and tested virtual machine

running most smart contract networks. Solidity is the programming language of choice for the EVM. It was created for this specific purpose. The main benefit of using EVM/Solidity is its sheer amount of resources from years of development. The IOTA Smart Contracts implementation is fully compatible with these resources, allowing to leverage all existing EVM developer tools for developing on the IOTA EVM. Any contracts previously written can be deployed on IOTA Smart Contracts without modification. Note that while EVM has become a standard for smart contract execution on L2s, it is not inherently designed for the unique features and capabilities of L1s like Move smart contracts. Move, with its focus on asset safety, resource management, and formal verification, offers a fundamentally different approach at the L1 level.

By combining these unique capabilities, IOTA delivers a resilient and future-ready infrastructure for diverse industries, from decentralized finance to supply chain management and beyond, all while setting a new standard for security, efficiency, and scalability in distributed ledger technology.

3.1. Object-Centric Design

The fundamental unit of storage on IOTA is the object. Unlike many blockchains that focus on accounts containing key-value stores, IOTA's storage model centers around objects, each of which is addressable on-chain by a unique ID. In IOTA, a smart contract is an object known as a package, and these smart contracts interact with other objects on the IOTA network:

- **IOTA Move Package:** A collection of IOTA Move bytecode modules. Each module has a unique name within its containing package. The combination of the package's on-chain ID and the module name uniquely identifies the module. When you publish smart contracts on IOTA, the package is the unit of publication. Once published, a package object can only be upgraded with the appropriate permissions. A package object may depend on other package objects previously published on IOTA.
- **IOTA Move Object:** Typed data managed by a specific IOTA Move module from an IOTA Move package. Each object value is a struct with fields that can include primitive types (such as integers and addresses), other objects, and non-object structs.

Transactions in IOTA take objects as input, modify these inputs, and produce new or updated objects as output. Each object records the hash of the last transaction that produced it as an output. This relationship between objects and transactions can be represented as a directed acyclic graph (DAG), where:

- Nodes represent transactions.
- Directed edges connect transaction A to transaction B if an output object of A is used as an input for B. These edges are labeled with the reference of the object, specifying the exact version created by A and used by B.

The root of this DAG is a genesis transaction that takes no inputs and produces the objects that exist in the system's initial state. The DAG can be extended by identifying mutable transaction outputs that have not been consumed and issuing a new transaction that takes these outputs (and optionally, immutable transaction outputs) as inputs.

The set of objects available as inputs to a transaction are the live objects, and the global state maintained by IOTA consists of all such objects. The live objects for a specific IOTA address A include all objects owned by A, along with all shared and immutable objects in the system.

When this DAG encompasses all committed transactions, it forms a complete and cryptographically auditable view of the system's state and history. Additionally, you can construct a DAG of the relevant history for a subset of transactions or objects, such as those owned by a single address.

Every object on the IOTA network has an ownership model that dictates how it can be accessed and used within transactions. The ownership type of an object determines who has the authority to interact with it and how it can be manipulated. In IOTA, objects can be:

- **Shared:** Accessible for reads and writes by any transaction
- **Owned:** Accessible for reads and writes by transactions signed by their owner.

Both shared and owned objects offer unique advantages and trade-offs that should be carefully considered when designing applications. Transactions involving only owned objects benefit from very low latency to finality because they bypass consensus. However, only the owner can access these objects, which complicates processes requiring interaction between multiple parties. On the other hand, transactions that involve shared objects require consensus to manage reads and writes, potentially leading to slightly higher gas costs and increased latency. When multiple shared objects or particularly popular objects are accessed, contention may further increase latency.

Below are examples of types of objects and their ownership in IOTA:

- **Immutable Objects:** In IOTA, objects can be either mutable or immutable. Immutable objects are unique in that they cannot be altered, transferred, or deleted after they are created. Once an object is made immutable, it has no owner, making it accessible to anyone on the network.
- **Address-Owned Objects:** An address-owned object is tied to a specific 32-byte address, which can either be an account address derived from a particular signature scheme or an object ID. These objects are exclusively accessible by their owner, meaning no other entity can interact with them unless ownership is transferred.
- **Shared Objects:** Shared objects are made accessible to everyone on the network through

the 0x2::transfer::share_object function. Unlike address-owned objects, shared objects are not restricted to a single owner and can be read or modified by any entity on the network. This extended accessibility can be advantageous for certain applications, but it also requires careful consideration of security, especially when sensitive operations are involved.

3.2. Consensus Mechanism

IOTA employs the Mysticeti consensus mechanism, a cutting-edge protocol designed to ensure security, scalability, and energy efficiency. Mysticeti operates within a Delegated Proof-of-Stake (DPoS) framework, where validators are selected based on stake to participate in the consensus process. This combination of technologies enables IOTA to achieve robust decentralization and network stability while maintaining high performance and sustainability.

Mysticeti utilizes a Directed Acyclic Graph (DAG) architecture to process transactions in parallel, achieving sub-second finality and supporting a throughput of over 50,000 transactions per second. Unlike traditional blockchain systems, Mysticeti eliminates explicit block certification by embedding consensus directly into the DAG structure, significantly reducing computational overhead and latency. Its leaderless design enhances fault tolerance by ensuring no single entity controls the process of transaction validation.

The Mysticeti protocol is underpinned by advanced cryptographic techniques, ensuring the security and integrity of the network. By leveraging the DPoS framework and the innovative properties of the DAG, IOTA achieves a scalable, efficient, and decentralized infrastructure suitable for powering secure, real-world applications across various industries.

3.3. Smart Contracts

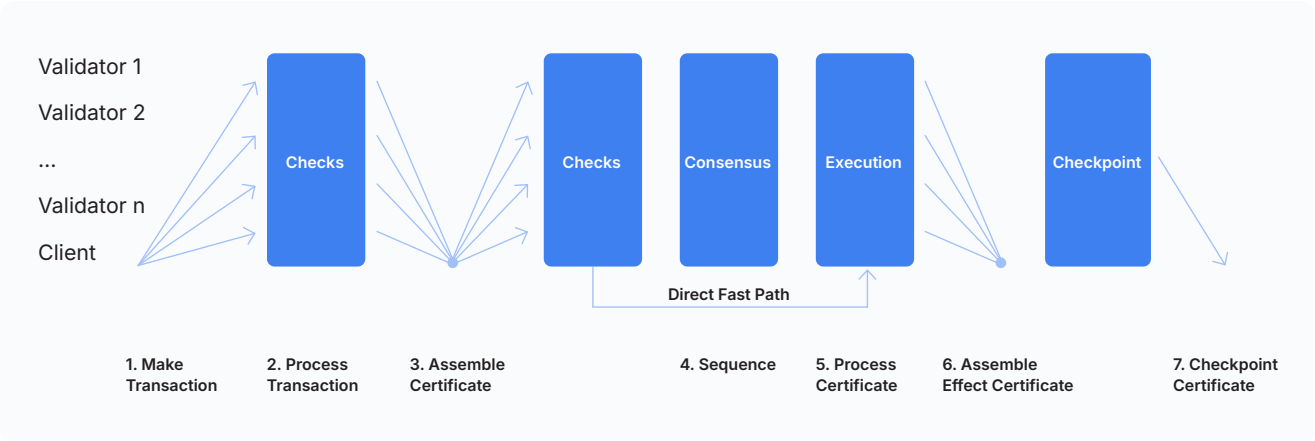
IOTA's baselayer protocol is designed not only as a robust distributed ledger but also as a flexible foundation that paves the way for additional technical standards and contract-based protocols. Central to this vision is the Move Virtual Machine (MoveVM), which provides a secure and resource-oriented programming environment. The MoveVM layer empowers developers to define and enforce new technical standards through smart contracts, ensuring that protocols built atop the IOTA baselayer can evolve alongside emerging technological and industry demands.

The IOTA protocol leverages the Move Virtual Machine (MoveVM) and the Move programming language to enable secure and efficient smart contracts and native asset management. Move's resource-oriented programming model ensures strict safeguards for the creation, transfer, and destruction of digital assets, preventing vulnerabilities such as double-spending and re-entrancy

attacks. This robust model supports the development of advanced decentralized applications (dApps) and tokenized ecosystems.

IOTA's ledger natively supports the creation and management of digital assets without relying on external layers or additional smart contracts. Developers can programmatically mint and burn assets with custom logic such as conditional transfers, vesting schedules, or compliance requirements.

Move smart contracts seamlessly interact with native assets, enabling secure and programmable functionalities for decentralized finance (DeFi), NFTs, and other tokenized applications. The resource-oriented design enforces asset integrity and simplifies complex workflows, ensuring efficiency and scalability.



IOTA's combination of Move and native asset capabilities delivers a secure, scalable platform for creating and managing digital ecosystems with minimal complexity and maximum flexibility.

3.4. Transaction Lifecycle

A high level overview of the transaction lifecycle is depicted and detailed below

- **Transaction creation:** A user submits a transaction to a full node and signs it with a private key. The transaction can affect objects owned by the user, as well as shared objects.
- **Submission to Validators:** The full node sends the transaction to the validator set. They check if it's valid and lock the involved "owned objects" to prevent double-spending. After validation, they send their signature back to the full node.
- **Transaction Certificate creation:** After collecting responses from a supermajority of validators, the full node can form a transaction certificate, which includes the transaction and the validator signatures. This certificate is sent back to all validators.
- **Consensus and Execution:** The validators verify the certificate transaction validity and begin the process required for transaction inclusion. If the transaction involves only owned

objects, IOTA can process and execute it immediately without waiting for consensus (fast path consensus). Otherwise, the transaction is submitted to IOTA's consensus and sequencing modules, and only later executed.

- **Effects Certificate:** After the transaction is executed, each validator signs off on the effects of the transaction and sends them back to the full node. The effects of a transaction are essentially a detailed record of what happened, including objects that were changed, created, wrapped, unwrapped, or deleted, the amount of gas spent, and the transaction's execution status. The full node then gathers these signed effects from a majority of validators, forming an effects certificate, which guarantees that the transaction is final.
- **Checkpoint inclusion:** The final stage in a transaction's life is its inclusion in a checkpoint. The consensus layer produces a globally agreed-upon ordering of transactions. Validators take chunks of this ordered list and use them to form checkpoints, containing a list of transaction digests and the digests of their effects. Once a transaction is included in a checkpoint, it becomes part of the permanent record on the IOTA network.

Appendix: Total and Circulating Supply

IOTA employs minting and burning mechanisms to dynamically regulate its circulating token supply. While there is no predefined upper limit on either the total or circulating supply, the design ensures that supply growth remains, at most, linear over time.

Total Supply Overview

The IOTA network launched (at the Rebase) with an initial total supply of 4,600,000,000 IOTA tokens. In each epoch—approximately one per day—up to 767,000 IOTA can be newly minted and distributed among participants. At the same time, every network transaction burns a small amount of tokens.

Assuming maximum daily issuance and no burning, the total supply could increase by up to 279,955,000 IOTA per 365 epochs (roughly one year), representing a maximum annual growth rate of 6.1% in the first year. In subsequent years, while the percentage increase will vary due to supply changes, the absolute annual supply increase remains constant.

Circulating Supply Overview

The circulating supply includes all tokens that are not locked at a given time. At the launch of the new IOTA mainnet on May 5, 2025, a portion of the initial total supply remained locked, in accordance with the IOTA Stardust and Tokenomics updates. At launch, the circulating supply stood at 3,746,223,720 IOTA.

In addition to the minting and burning mechanisms, the circulating supply will increase until October 2027 due to scheduled token unlocks. From that point onward, total supply and circulating supply will be equal, as all tokens will have been unlocked.

Unlocking Schedule

May 14 – October 1, 2025: Every two weeks, 19,137,719 IOTA will be unlocked, totaling 210,514,906 IOTA by October 1. Including token minting, the maximum circulating supply at that point will be approximately 4,074,089,626 IOTA.

October 2025 – September 29, 2027: The biweekly unlocking amount reduces to 12,370,411 IOTA, resulting in an additional 643,261,373 IOTA being unlocked over this period. Including continued minting, the maximum circulating supply at the end of this period will reach 5,274,192,999 IOTA.

A graph below illustrates the projected maximum total and circulating supply over time. This trend can be extrapolated beyond 2027, as token issuance continues at a consistent rate.

IOTA Maximum Supply (millions of IOTAs)