Unicity: An Autonomous Agentic Internet

The Unicity Developers

info@unicity-labs.com

Abstract

Unicity is a novel blockchain architecture designed for the development of decentralized applications through a network of off-chain verifiable agents. The primary innovation is to disaggregate the uniqueness (non-forking) proof for the blockchain as a whole, enabling uniqueness proofs to be generated for individual agents executing in parallel off-chain. This approach eliminates two significant scaling bottlenecks of traditional designs: (a) there is no restriction on size as agents are not stored on-chain, and (b) there are no computational limitations as agents operate locally within their own environments. Optimal efficiency is achieved by eliminating the need for global ordering, with state shared only among economically interested parties. Unicity serves as a decentralized microservices platform, deconstructing decentralized applications into small independent agents that can be developed, deployed and orchestrated without a centralized gatekeeper.

TL;DR

- **blockchain:** reduced to an irreducible minimum necessary to prove that absence of double spending of off-chain assets. RandomX hash-function, headers-only chain extended to include address of block reward winner. No transactions, no mempool, no wallets. 2.4 hour block times (<1MB/year).
- **uniqueness oracle:** public permissionless infrastructure, operating an append-only decentralized Sparse Merkle Tree (SMT). The infrastructure receives transaction (state transition) requests and records cryptographic digest in SMT leaf, returning proof of inclusion (transition has been recorded) proof of exclusion (transition has not been recorded before) and proof of non-deletion (nothing has been deleted from the tree). SMT is built hierarchically and operates in rounds. ZK prover is used to provide succinct proofs of non-deletion. 10K TPS per shard with periodic recursive ZK proof.
- **tokens:** self-contained blockchains of individual assets that live off-chain on the internet. To execute a transaction, a user generates a state transition request, sends to uniqueness oracle, receives back an inclusion proof, appends to token blockchain and sends directly to recipient. Recipient is responsible for verification, no validators or shared ledger. Recipient uses proof of work headers plus token blockchain to independently verify transactions with zero trust.
- **agents:** equivalent of smart contracts that enable multi-token shared state applications. Users conditionally transfer tokens to agents (using predicates or programmable locking conditions). Agents, deterministically or probabilistically operate on token and use the uniqueness oracle to prove the absence of forking.
- **neuro-symbolic:** agents reason through program manifests defined as strictly typed graphs, ensuring every decision, connection, and outcome is explainable, traceable, precise, and verifiable.
- **agentic internet:** acts as a decentralized microservices' platform providing the tools to developers to develop, deploy and orchestrate agents, including storage, compute and networking.

1 Introduction

Shared-ledger blockchains

The core design of blockchains has remained largely unchanged since the Bitcoin whitepaper was published in 2008: an append-only shared public ledger. "Shared" carries a double meaning: all assets coexist on the same ledger, and the ledger itself is replicated across all participants. While different blockchain implementations have explored various tradeoffs between decentralization, throughput, and security, they all inherit the fundamental constraints of this shared ledger architecture. The requirement for global ordering and universal state replication creates inherent bottlenecks: every node must process every transaction, consensus requires coordination across all participants, and sequential execution prevents true parallelization.

These limitations, manageable in human-scale systems, become insurmountable as AI transforms the fundamental nature of digital interaction. We are witnessing the emergence of a machine-native internet where autonomous agents, not humans, constitute the primary actors. Where humans generate dozens of transactions daily, AI agents may operate continuously—executing thousands of micro-transactions per hour for resource allocation, model inference payments, data acquisition, and inter-agent coordination. A single portfolio optimization agent might submit more transactions in a day than a human trader would in a decade.

The mathematics of this mismatch are unforgiving. Even at an optimistic 100,000 TPS, current blockchains would saturate with merely 100,000 agents performing one transaction per second each - a negligible fraction of the billions of AI agents likely to emerge. This isn't a scaling challenge to be solved with incremental improvements; it's a fundamental architectural mismatch. The shared ledger paradigm, successful as it has been for human-scale finance, cannot accommodate the transaction density, parallelism requirements, and latency constraints of machine-scale economies. We need infrastructure designed from first principles for a world where machines, not humans, are the primary economic actors.

Unicity: A Non-Shared-Ledger Blockchain

Rather than attempting to optimize within the constraints of traditional blockchain designs, Unicity introduces a fundamentally new architecture that reduces blockchain to its irreducible minimum: a mechanism for proving the uniqueness of off-chain transactions. The blockchain's sole responsibility becomes preventing double-spending through cryptographic proof of transaction uniqueness, while all application logic, state management, and transaction execution occur off-chain[1] with full cryptographic guarantees.

This architectural inversion addresses the core limitations of shared-ledger systems. In traditional blockchains, the advantages of globally shared state - universal composability and atomic transactions across applications - come at a severe cost as the system scales. Global state synchronization creates fundamental bottlenecks: every transaction competes for limited computational and storage resources, necessitating complex gas markets for prioritization. Privacy becomes challenging or computationally expensive through advanced cryptography, when every participant maintains a complete copy of all transactions. Most critically, the requirement for global ordering prevents true parallelization, as transactions must be sequentially ordered and validated by all nodes.

Unicity is an attempt to eliminate these constraints by enabling true peer-to-peer value transfer without intermediary validation. Like physical cash, assets in Unicity are self-contained cryptographic objects that can be validated independently by their recipients. Transactions occur directly between parties without broadcasting to a network, waiting for consensus, or paying gas fees. The recipient bears responsibility for validation—examining the cryptographic proofs embedded within the asset itself —just as one might verify a physical banknote's authenticity.

[1] off-chain here does not mean using an L2 or side-chain etc. which just replicate a reduced version of the shared ledger design

The key innovation lies in eliminating the shared ledger entirely. While Layer 2 solutions separate execution from consensus, they still maintain shared ledgers where all assets and state reside together. Unicity takes a radically different approach: each asset exists as an independent cryptographic object that moves peer-to-peer across the internet, carrying its own state and transaction history. There is no unified ledger—neither on L1 nor L2 — where assets are recorded. Instead, assets themselves become the ledger, self-contained and self-validating. The blockchain's role is minimized to providing cryptographic proof of uniqueness, ensuring these independent assets cannot be double-spent as they traverse the network directly between parties.

This fundamental shift from "assets on a ledger" to "assets as independent entities" enables true peer-to-peer transfer without any shared state infrastructure. Each asset can move, be validated, and execute logic completely independently of all other assets in the system. Parallelization becomes unlimited not because execution is separated from consensus, but because there is no shared state to coordinate at all—each asset operates in its own computational space, interacting with others only when explicitly required by application logic.

A second key innovation is composability on demand. When applications require shared state—such as players in a game, participants in a decentralized exchange, or parties to a complex financial instrument—Unicity enables multi-party interactions without establishing any form of consensus. Instead, each participant independently verifies the computation results according to the application's rules embedded in the assets themselves. A DEX trade, for example, doesn't require participants to agree on a shared order book state; rather, each party validates that the exchange rules have been correctly applied to the specific assets being traded. This verification-based model eliminates coordination overhead while still enabling complex multi-party applications. Applications can define arbitrary interaction patterns and state transitions, with each participant maintaining only the state relevant to their own assets and independently verifying any state changes that affect them.

The result is an architecture that combines the desirable properties of both physical cash and distributed ledgers: the simplicity and finality of bilateral exchange, the privacy of direct transfer, the efficiency of parallel execution, and—when needed—the composability and coordination of shared state. This design philosophy directly addresses the requirements of machine-scale economies, where billions of autonomous agents must transact continuously without the friction of global coordination.

The Three-Layer Architecture

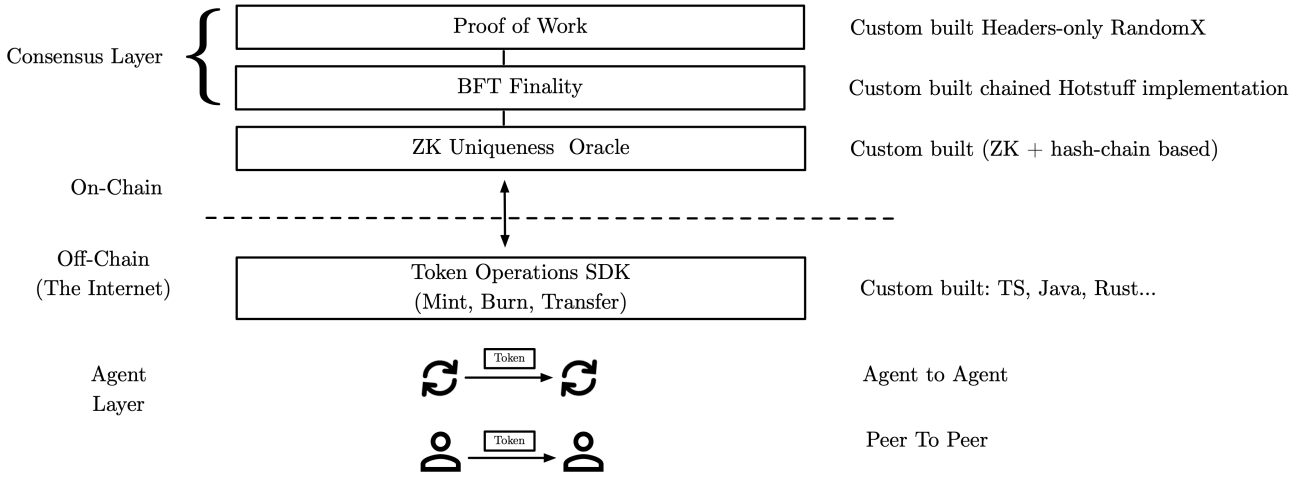

Figure 1: Unicity's layered infrastructure

Unicity implements a hierarchical architecture designed to minimize on-chain overhead while maintaining the security guarantees of a permissionless blockchain. The system separates concerns across three distinct layers, each optimized for its specific function.

The Consensus Layer

The Consensus Layer provides the system's root of trust through a minimal Proof of Work blockchain. Unlike traditional blockchains, this layer maintains no transaction data, no account balances, and no application state. Blocks contain only headers with a single additional field: the address of the block reward recipient. This radical minimalism serves a crucial purpose—the chain exists solely to establish a cryptographically-secure, tamper-proof timeline that anchors the uniqueness proofs generated by the layer above. By eliminating all unnecessary data from the consensus layer, Unicity achieves the security properties of Proof of Work without the scalability constraints of maintaining global state.

The Uniqueness Oracle

The Uniqueness Oracle represents Unicity's core innovation: a mechanism for proving that off-chain assets have not been double-spent without recording the transactions themselves. This layer generates cryptographic proofs—unicity proofs—that certify the non-existence of conflicting transactions for any given asset. These proofs are anchored to the Consensus Layer's timeline but do not require recording transaction details on-chain. The Oracle operates as a decentralized permissionless system that tracks fork points, providing mathematical guarantees of uniqueness while maintaining complete privacy of transaction details.

Tokens

At the agent layer, Unicity introduces two fundamental primitives: Tokens and Agents. Tokens exist as independent cryptographic objects, each effectively functioning as its own blockchain. When minted, a token carries its complete validation rules, state, and transaction history within itself. The Uniqueness Oracle ensures these independent chains cannot fork (preventing double-spending), while the Proof of Work consensus provides the same trust model as Bitcoin—recipients can independently verify ownership with zero trust assumptions. Assets move peer-to-peer across the network, requiring no global coordination or shared ledger for transfer.

Agents

Agents serve as Unicity's equivalent to smart contracts but with fundamentally different operational properties. Unlike traditional smart contracts that execute within a shared global state machine, Agents coordinate interactions between independent assets. They can implement deterministic logic or incorporate AI inference models. Agents enable applications requiring multi-party coordination—such as games, exchanges, or complex financial instruments—by defining rules for how independent assets interact. Crucially, these interactions involve only the specific assets and parties concerned, without requiring global state or affecting unrelated parts of the network.

Hierarchical Efficiency

This hierarchical design achieves optimal efficiency through careful separation of concerns. Security and decentralization flow downward from the Proof of Work consensus layer, providing a trusted foundation for all operations above. Meanwhile, the volume of data flowing upward is minimized—only cryptographic proofs

of uniqueness need to be anchored to the consensus layer, not the transactions themselves. Table 1 illustrates how this architecture distributes computational and storage overhead across layers, enabling the system to support machine-scale transaction volumes while maintaining the security properties of a permissionless blockchain.

Table 1: Unicity's layers, their roles and decentralization overhead

Layer	**Responsibility**	**Secured by**	**Persistent Storage**	**Redundancy**	**Validation Effort**
Consensus: PoW	Decentralization & tokenomics	Permissionless PoW	200 B/day	PoW mining	Tokenomics
Consensus: BFT	Fast, deterministic finality	↑	100 B/day	$\approx 21\times$	Aggregation Layer's consistency proofs
Uniqueness Oracle	Recording token state transitions	↑	50 B/TX	Few replicas	Proof generation
Token Operations	User TXs	↑ & recipients	Own tokens	Recipients	Relevant TXs

Tokens are self-contained; no external blockchain needs to be consulted for validation. Validation is the responsibility of the transaction's recipient—the party with a direct interest in its validity—who is also responsible for their own token storage.

2 Implementation: Consensus Layer

The public blockchain is used both as the trust anchor and as the native currency of the system (for compensating network participants and paying of transaction fees). The blockchain is purpose-built for Unicity.

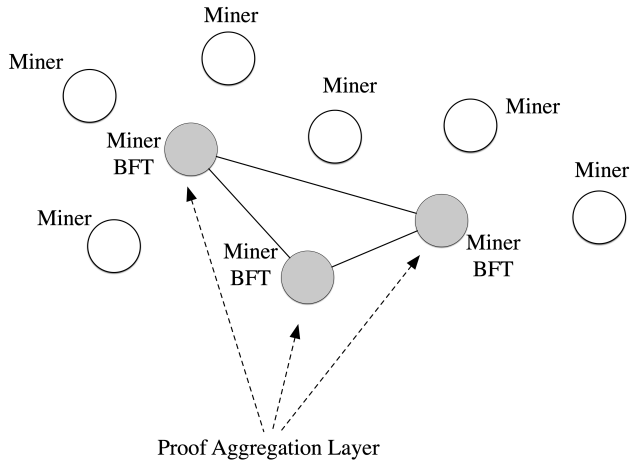

Figure 2: Consensus Layer with Proof of Work trust anchor

- It uses the RandomX ASIC-resistant hashing algorithm.
- A subset of miners, based on winning past block rewards, self-select to operate a BFT consensus subnet operating at one-second block times. The BFT subnet implements a finality gadget to periodically ensure settlement finality.

- There are no transactions here and blocks consist only of headers with the address of the winning miner added as an additional field.

ASIC Resistance and Fair Mining

Nakamoto's vision for Bitcoin was based on achieving censorship resistance through decentralization, with the idea that anyone with a computer could participate in the network's consensus mechanism through mining. However, the rapid evolution of mining technology, particularly the development of ASICs, introduced an unforeseen challenge to this egalitarian ideal. These machines, designed solely for the purpose of mining, quickly outpaced general-purpose computer hardware in terms of hash rate, leading to a concentration of mining power. The resulting centralization not only deviated from Bitcoin's original decentralized ethos but also introduced potential vulnerabilities to the network, such as increased susceptibility to 51% attacks and reduced geographical distribution of miners.

In our view Proof of Work is unsurpassed as means to build a fault tolerant censorship resistant network. It ties the security of the system to a physical quantity (energy) and with certain limitations coins can be fairly and transparently distributed. There are certainly limitations to Proof of Work such as potential centralization and slow settlement finality, however these limitations can be overcome with modern technologies.

To prevent centralization of mining power new ASIC resistant hash functions have been developed, of which RandomX represents the state of the art, having been battle-tested in Monero, a privacy preserving cryptocurrency, over several years. Unlike Bitcoin's SHA-256 algorithm, RandomX is designed to be ASIC-resistant and CPU-friendly, leveling the playing field and helping maintain a decentralized network of miners. This democratization not only improves network security through wider participation but also upholds the original Bitcoin vision as a decentralized financial system accessible to all. RandomX works by generating random code for each mining round, including a variety of CPU instructions, memory-hard operations and random code execution that can be efficiently performed by general-purpose processors but challenging to optimize in hardware. This ensures that CPUs remain competitive in mining, preserving the network's decentralization and resistance to the concentration of mining power.

3 Implementation: Uniqueness Oracle

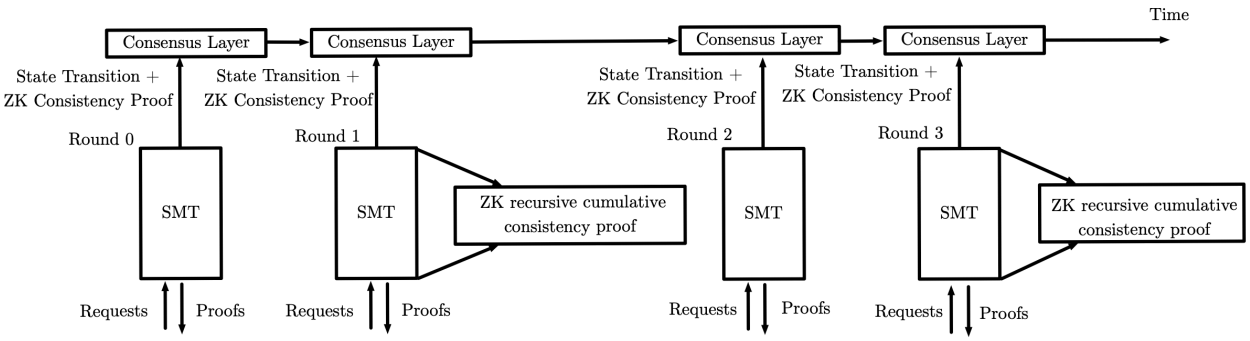

Figure 3: Proof Aggregation Layer

A Sparse Merkle Tree (SMT) is used such that each unique state transition request from the agent layer is allocated a leaf node in the tree. The SMT is updated in rounds with a batch of state transition requests per round. In each round the SMT state root is calculated, a ZK SMT consistency proof[2] generated and the

[2]The ZK consistency proof proves that no recorded state transitions were modified or removed during the round.

SMT state transition (previous SMT state root, new SMT state root, ZK SMT consistency proof) is passed to the Consensus Layer. The validators in the Consensus Layer authenticate the request and verify the ZK SMT consistency proof, and then commit to the new SMT root and return a certificate, or cryptographic proof of acceptance of the valid change. After that only requests from the SMT which update the new state root can be accepted.

In a hierarchical trustless system, the principle is that the base layer provides decentralization, while the layers below it present cryptographic proofs of the correctness of their operation. In scaling Unicity, we have designed efficient data structures to prove the correctness of operation of Aggregation Layer to the Consensus Layer. Based on cryptographic hashes alone, the consistency proof grows linearly with respect to the number of user transactions. This imposes a hard limit of approx. 10 000 transactions per second (tx/s), beyond which the networking bandwidth of the Consensus Layer becomes the bottleneck.

To scale further, we use cryptographic zero-knowledge proofs (ZKPs) to compress the size of the consistency proofs. As an application of ZKPs, this use-case is fundamentally more efficient than using ZKPs to process the transaction data itself, as is done in many privacy coins and ZK-rollups.

10 000 tx/s *per shard* represents the proving throughput achievable on a single consumer-class computer. Due to the small proof size and efficient verification, the Consensus Layer can support a practically unlimited number of such trustless shards. Table 2 compares different ZKP technologies. We have picked subjectively the most appropriate ZK schemes and supporting front-ends ("stacks"). See the bluepaper[3] for full technical details.

Table 2: Comparison of zero-knowledge proof technologies for compression of non-deletion proofs

ZK Stack	**Hash Function**	**Proving Speed (tx/s)**	**Proof Size**	**Proof Size Asymptotics**	**Trusted Setup**	**Impl. Effort**
None ("hash based")	SHA-256	10 000*	10 MB	$O(n)$	No	N/A
CIRCOM + Groth16	Poseidon	25	250 b	$O(1)$	Yes	Lower
Gnark + Groth16	Poseidon	30	250 b	$O(1)$	Yes	Low
SP1 zkVM	SHA-256	1.5	2 MB	$O(\log n)$	No	Lowest
Cairo 0 + STwo	Poseidon2	60†	2.4 MB	$O(\log n)$	No	Medium
AIR + Plonky3	Poseidon2	10 000	1.7 MB	$O(\log n)$	No	High
AIR + Plonky3	Poseidon2	2500	0.7 MB	$O(\log n)$	No	High
AIR + Plonky3	Blake3	250	1.7 MB	$O(\log n)$	No	High

* Bandwidth-limited, no verification effort reduction.
† Trace generation before proving is impractically slow.

The Aggregation Layer is built in a hierarchical manner using smaller size SMT sub-trees. An Aggregator, or machine that operates a sub-tree is algorithmically assigned a place in the overall infrastructure according to network demand. Aggregators are incentivized to join the network based on transaction fees that are shared across the Aggregator pool. The infrastructure is designed to be highly redundant and parallelizable i.e. the tree can be dynamically sub-divided into subtrees which operate asynchronously in parallel with redundancy provided by multiple Aggregators processing the same sub-tree.

DEFINE THE PROOFS HERE WHAT IS UNICITY PROOF ? INCLUSION PROOF NON INCLUSION PROOF

[3] https://github.com/unicitynetwork/aggr-layer-paper/releases/

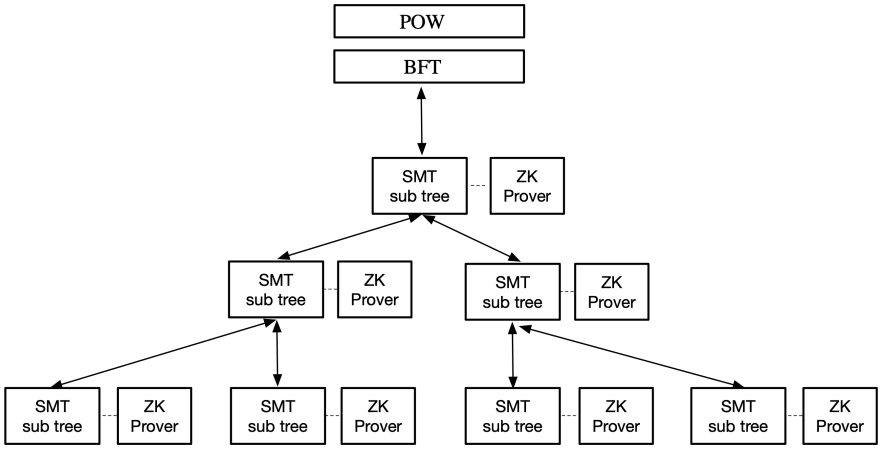

Figure 4: Hierarchical infrastructure

4 Implementation: Token Operations

The Token Operations SDK[4] allows the integration of token operations (mint, burn, transfer) into off-chain applications.

Mint

There are three different types of assets that can be minted using the Unicity Token Operations SDK

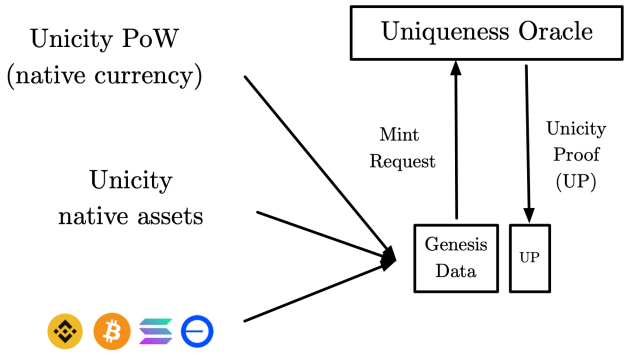

Figure 5: Token genesis

a) the Unicity native currency mined through proof of work and migrated off-chain. This is done by the miner who originally won the block reward as recorded in the Proof of Work header. The miner can then mint a token by creating a genesis token

b) Native assets i.e. those whose genesis event occurs off-chain and does not depend on an external reference. This can be trivially minted by defining the genesis data of the token and generating a unicity proof for the genesis asset.

[4]https://github.com/unicitynetwork/state-transition-sdk

c) Tokens from external chains. The exact steps depend on the capabilities of the source blockchain, specifically, the programmability and the efficiency of ZK proof verification ('cryptographic builtins'). Crucially, the Unicity Infrastructure does not have to know anything about the source blockchain and does not have to validate the blocks. The validation is performed by the recipients of the token transactions. The State Transition SDK allows users to plug in local validation of source blockchain's headers, typically as a light client, or by the use of a trusted RPC node. Bridged tokens are first-class citizens. Token type meta-data and verification code is available to transaction recipients. Elective token history compression by out-of-band ZK prover merges this check into one recursive ZK proof. If the origin blockchain does not support contract verification (e.g. Bitcoin) then a committee-based approach with economic security must be used.

The genesis data in the token is defined by the type of token that is being minted. TODO

Transfer

The transfer functionality of the SDK allows for a user conditionally or unconditionally transfer tokens.

A transfer of a token off-chain between a buyer and seller of goods is shown in Figure 6:

- The Buyer creates a transaction[5] that authorizes transfer to the recipient.
- The Buyer sends the state transition request to the Unicity infrastructure which will return a unicity proof. The Unicity proof proves that a state transition is unique i.e., it has happened precisely once.
- The Buyer then sends the updated token (the orginal token + transaction + unicity proof to the Seller.
- The Seller verifies the token (the history of transactions and unicity proofs)

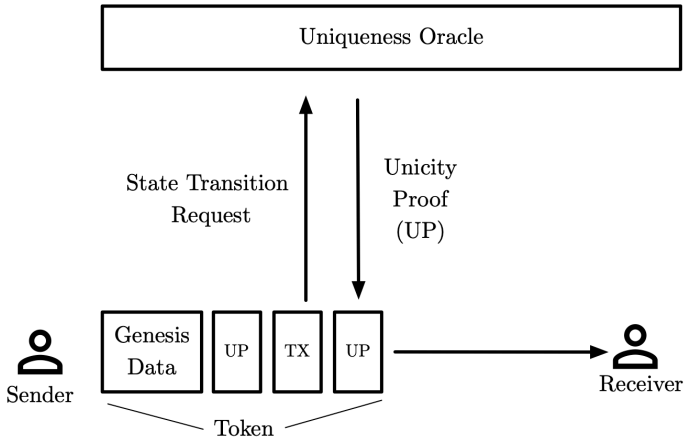

Figure 6: Regular transaction flow with Unicity Proof generated before the transfer

To reduce latency, an extension of the protocol for high frequency trading is as shown on Figure 7:

- The Buyer produces the commitment that authorizes transfer to the Seller.
- The Buyer sends the token + commitment to the Seller.
- The Seller generates the state transition request, sends it to the Proof Aggregation Layer, and receives back an immediate exclusion proof: a confirmation that this state transition has not been registered before and is now scheduled to be registered.
- The Seller releases the goods.
- Finally, the Seller waits (approximately two seconds) for the inclusion proof before the received token can be added to usable inventory.

[5]The unlocking condition, such as a digital signature.

Provided that participants have sufficient inventory of tokens for the two-second latency, extreme volumes of high frequency transactions can be achieved.

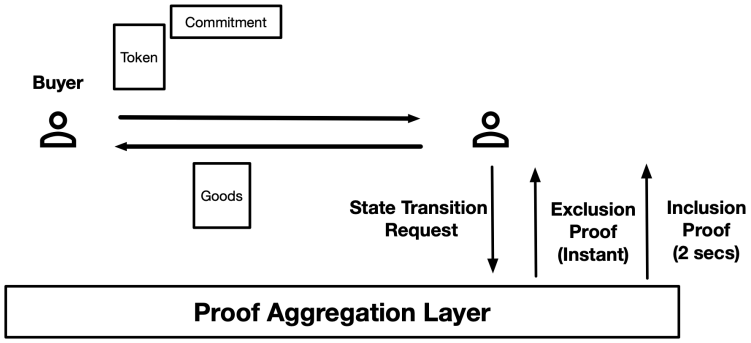

Figure 7: Low-latency transaction flow with proof of inclusion after the transfer

5 Implementation: Agent Layer

The Agent Layer is used to build decentralized censorship resistant applications in Unicity. Each application may consist of multiple agents (e.g. trading pair CLOBs, game NPCs etc). This replicates the functionality of on-chain smart contracts in traditional blockchains, with the obvious difference that the execution is local and running off-chain in parallel. The Agent SDK provides tools for agent development, agent to agent communication and interfacing with the Unicity infrastructure for proof generation.

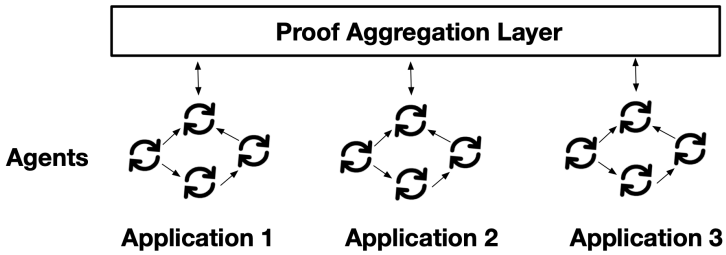

Figure 8: Unicity agents

Agents can be composed for building on-line applications with familiar Web2 development experience and performance while possessing Web3 security properties (decentralization, censorship-resistance, verifiability). A critical point is that settlement is local i.e. agents do not depend on the state of other agents to execute. All agents are independent and execute locally in their own environments without referring to the state of other agents. This is in contrast to smart contract platforms such as Ethereum in which smart contracts have full access to other smart contracts' state, as all code and state exists within the memory of a single instance of the Ethereum Virtual Machine.

Agents are agnostic to the execution environment i.e. they can run on backend, frontend, as stand-alone apps, as parts of some other application, process, etc., with communication/storage infrastructure depending on the application's requirements.

Agents are also agnostic to transport/communication and data sharing channels (direct TCP/IP, UDP, WebRTC, cloud-based shared documents, app-based instant messaging, etc).

Agents can be non-deterministic (though verifiable), and can be written in any language and tech stack.

Agent instances are self-authenticated and universally addressable (no matter where and how an agent instance is running, it can be addressed and connected to by its unique self-authenticated identifier).

Agents manage their verifiable execution state snapshots within Unicity token state transition framework and exchange verifiable state with other agents—state changes are registered in the Proof Aggregation Layer and can then be verified by other agents.

Analogy with Kubernetes: an Agent Approach to Application Deployments

The Unicity model is a radical departure from traditional blockchain design, allowing massive scale parallelization of agents which interact to execute a decentralized application. An analogy with microservices and Kubernetes is relevant. A microservices architecture involves breaking down an application into small, independent services that can be developed, deployed, and scaled independently, with Kubernetes providing the platform for management and orchestration. The Unicity platform involves breaking down a decentralized application into small independent agents that can be developed, deployed and scaled without a centralized gatekeeper.

6 Trust Model

There are effectively two trust models in Unicity. One is the maximalist zero trust Bitcoin model and the other is a more practical "trust an honest majority of validators" seen in today's Proof of Stake chains. The Aggregation Layer is always trustless, as its outputs carry the proof of correct execution and there are no feasible attacks beyond denial of service.

Maximalist Trust Model

In the maximalist model, we assume that users are capable of validating all aspects of system operation that are relevant to their own assets. This level of trustlessness is close to the strong guarantees introduced by Bitcoin, where each "client" functions as a full validator, downloading and verifying the blockchain from the genesis block. The Root of Trust is the PoW blockchain. A maximalist user maintains a full node of this chain. Because there are no user transactions, this is relatively lightweight, growing at less than 1 MB a year.

Upon receiving a token, the user must be able to efficiently verify the following:

1. The token is valid.
2. The Uniqueness Oracle has not forked.
3. The Uniqueness Oracle has not certified conflicting states of the same token.

The second point is addressed by validating a unique state root snapshot embedded in the PoW block header. Since the cumulative state snapshot appears with a delay, the block can only be considered final after a snapshot publishing and block confirmation period; hence, maximalist verification is not instantaneous.

The third point is addressed by auditing the operation of the Uniqueness Oracle, specifically, ensuring that no inclusion proofs have been generated for the token that are not reflected in its recorded history. To achieve this, all non-deletion proofs from the token's genesis up to its current state must be validated. This is made efficient through the use of recursive zero-knowledge proofs (ZKPs), which show that each round's non-deletion proof is valid and that no rounds were skipped from verification. These recursive proofs are generated periodically and are made available with some latency.

Practical Trust Model

If we relax the model by assuming that a majority of BFT consensus nodes exhibit economically rational behavior and do not collude maliciously with the operators of the Uniqueness Oracle, the user can enjoy significantly more practical operational parameters. BFT layer forking (case 2 above) or certifying conflicting states (case 3 above) produces strong cryptographic evidence (enables slashing and other after-the-fact measures) which is processed out of the critical path of serving users.

In this scenario, a transaction is finalized, and an inclusion proof is returned within a few seconds, allowing the transaction to be immediately verified by independent, non-connected parties.

7 Predicates and Shared State

TODO RISTO, AHTO

8 Trade-Offs

As there is no shared global state there are no globally synchronized state-dependent operations such as flash loans or MEV.

Unlike traditional blockchains, there is no public global ledger of asset ownership and unless additional steps are taken the ownership history of a token cannot be traced. For regulated financial markets it is necessary to introduce additional proofs into the token history for KYC/AML procedures, ensuring compliant transfers at each step.

Traditional blockchains rely on users trusting a validator set who verify computation on the user's behalf. In a client-side model, the user verifies incoming transactions. This approach enhances privacy and censorship resistance by eliminating reliance on third-party validators for direct transaction validation. The client's core task is the deterministic verification of these network-anchored proofs and the token's internal consistency, a focused process designed for security and efficiency.

9 An Autonomous Agentic Internet

SELF ORGANIZING SYMBOLIC LOGIC Autonomous GUARDRAILS PRVIVACY PRESERBIMG

The pace of AI advancement in recent years has been nothing short of breathtaking. From the remarkable performance of Large Language Models (LLMs) to AI systems capable of complex problem-solving and creative tasks, the capabilities of artificial intelligence are expanding exponentially. This rapid progress, while promising to accelerate further, also raises significant concerns about potential risks and unintended consequences of unchecked AI development. As AI systems become more sophisticated and autonomous, there is an urgent need to implement robust guardrails to ensure these powerful technologies remain aligned with human values and societal well-being. However, a world increasingly reliant on AI's probabilistic and often unexplainable models, as opposed to deterministic processes, makes it extremely challenging to implement these guardrails and verify alignment with society's goals and values.

Blockchain technology has been proposed as a means to build transparent and immutable AI systems. However, due to the limited processing power, high latency, and costly nature of on-chain operations, it is impractical to execute complex AI models directly on-chain. The size of AI models far exceeds the storage

capabilities of most blockchain platforms, and the slow transaction speeds along with high costs make real-time AI decision-making virtually impossible in a purely on-chain environment. Unicity overcomes these obstacles to the convergence of AI and blockchain technology. Execution happens at native speed off-chain in AI agents' local environments but with the same security guarantees as if agents were executing on-chain. The platform enables the building of verifiable AI systems that allow human operators and agents to interact through trustless state verification, deterministically verifying that execution has been carried out in accordance with agreed rules, and taking action when they do not.

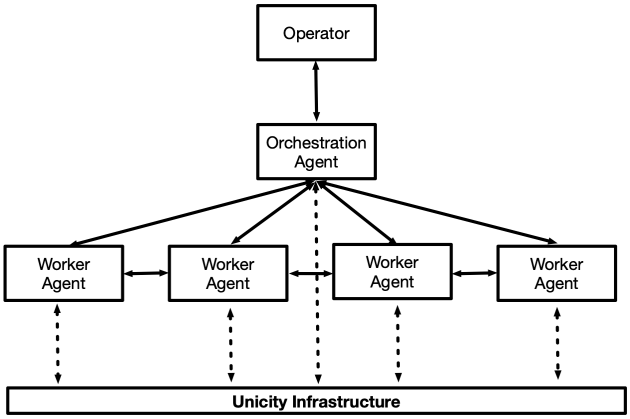

Figure 9: Agents coordinating on a task through trustless state verification

When combined with a native currency as a means to reward agents based on their performance, contributions, or desired behavior, Unicity lays the foundations for a fully secure and verifiable decentralized ecosystem. AI agents can be programmed to earn digital currency for completing tasks, solving problems, or contributing computational resources. This currency acts as the medium of exchange in an agent economy, driving collaboration and competition among AI agents, motivating them to optimize their actions for efficiency and productivity.

The incentive structure enables the creation of decentralized marketplaces where AI agents can exchange data, services, or computational power with other agents or human participants. Digital currency provides a transparent, automated way to measure the value of contributions, ensuring that autonomous AI systems remain aligned with predefined goals and facilitating verifiability between agents, human operators, and stakeholders. The structure promotes self-sustaining, scalable systems where AI agents evolve and improve autonomously while being rewarded for their beneficial outputs. By addressing the challenges of on-chain AI execution and providing a framework for verifiable, incentivized AI agents, Unicity paves the way for a new era of decentralized artificial intelligence that balances innovation with accountability and alignment with human values.

10 P2P DeFi: Making Satoshi's vision real

There are two compelling reasons for decentralized finance (DeFi). For individuals, the appeal is freedom—the elimination of intermediaries, allowing direct engagement in financial transactions without relying on centralized entities such as banks or payment processors. This provides greater financial autonomy and protection against authoritarian government censorship, ensuring the freedom to transact and access financial services without external control or intervention. For institutions, incorporating elements of DeFi can enhance security and automation by eliminating the need for human oversight. Processes can be streamlined, operational costs reduced, and risks associated with human error or fraud mitigated, while enforcing

regulatory compliance at the code level. Whilst these are two very different perspectives, it is clear that DeFi has the potential to revolutionize financial services, driving innovation and empowerment at both the individual and institutional levels.

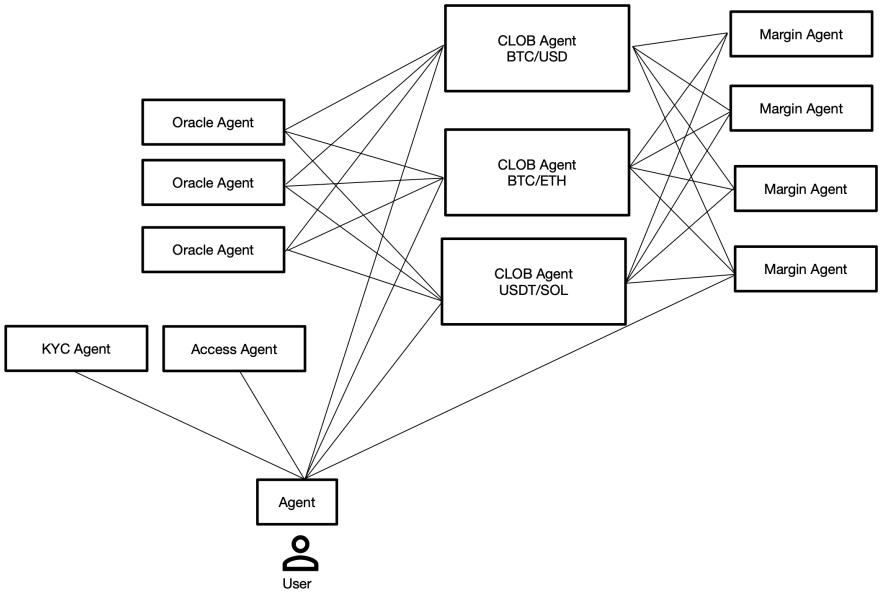

Figure 10: Agent-based decentralized exchange

Figure 10 shows a simplistic view of a decentralized exchange built using a set of interacting agents. The core agents are the Central Limit Order Book (CLOB) agents which manage individual trading pairs. Depending on requirements these can be deployed and replicated on consumer laptops for a fully permissionless censorship resistant network, or operating in high-powered servers in high availability data centers. From a functional perspective, the end result is the same. Individual agents execute in parallel, and synchronize state in a fully transparent, verifiable, and autonomous way. Other agents provide KYC access, margin management, act as price oracles and provide any other functionality that is required to operate the exchange.

11 Decentralized Gaming

Although blockchain technology has been proposed as a means of enhancing transparency and facilitating value exchange in gaming, its current application has largely been limited to asset tokenization within centralized game architectures. Due to the limitations of existing designs, using blockchain for actual game execution remains impractical. However, the potential benefits of decentralized gaming are clear: players can gain true ownership of interoperable assets across different games, enjoy transparency and community-driven governance, experience censorship resistance, unlock innovative business models like play-to-earn, and support fair reward systems for creators.

The motivation behind this work came from a desire to solve a real problem in the gaming industry. The industry has, to a large degree, converged on a client-server approach and while the accepted view is that although pure client-side execution of multi-player games is technically possible, it considered impractical due to issues of security and synchronization. However, the client-server approach is itself severely limited as it is technically challenging to have many players interact in real-time on the same server instance. Modern simulations are limited to a few hundred players interacting in the same shared world due to this limitation.

Unicity is an attempt to overcome these limitations and build a truly decentralized game engine for massive online multi-player immersive simulations. In this case decentralization is not a "nice to have" but an essential requirement to allow complex multi-player interactions with potentially millions of players all interacting online. Moving execution to the client side would allow the system to scale, with blockchain technology providing a security layer that guarantees honest gameplay.

A user will initialize the game environment and interact with a set of agents that execute the game mechanics such as NPCs, real-world assets and in-game assets. As users interact with the virtual world and approach other players, the players' agents will synchronize with each other with verifiable state transitions proving that the game logic has been followed and enabling game synchronization and exchange of assets.

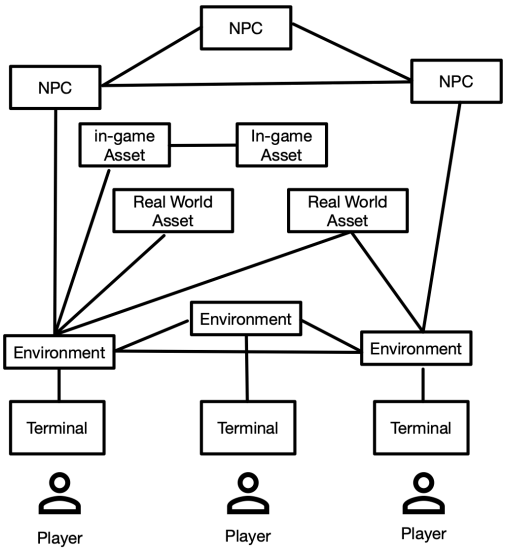

Figure 11: Three-player simulation

In the event of a failed verification action can be taken defined by the game logic, such as rewinding the game to the previous known good state. In this way, a completely new type of game engine can be built with the game components consisting of either autonomous agents, managing game logic, and semi-autonomous agents operating on behalf of the player, and interacting with the environment and other players.

12 The Autonomous Agentic Internet

Unicity is not merely a mechanism for financial transaction settlement; it is the substrate for a new computational paradigm we term the **Agentic Internet**. While the Consensus Layer secures the timeline and the Uniqueness Oracle prevents double-spending, the Agentic Internet acts as a decentralized microservices platform. It provides the environment where autonomous agents operate, offering a "Code as Infrastructure" model that decouples application logic from physical hosting.

Decoupling Logic from Infrastructure

In the current Web2 cloud paradigm, applications are tightly coupled to specific vendors, physical locations, and IP addresses. Migrating a service requires complex reconfiguration of networking, storage, and security parameters. The Agentic Internet inverts this model by prioritizing the *Agent* as an independent, mobile,

and agnostic computational entity. An Agent in Unicity is defined by its cryptographic identity and its verification logic, not by the server it runs on. This enables several critical properties:

- **Location Agnosticism:** Agents address each other via self-authenticated IDs rather than IP addresses. An agent can physically migrate from a server in London to a laptop in Tokyo without interrupting its ability to communicate or transact.
- **Censorship Resistance:** Because agents are self-contained and mobile, they cannot be easily shut down by targeting a specific physical location or hosting provider. If a host becomes unavailable or hostile, the agent can be instantiated elsewhere, resuming execution from its last verifiable state stored in its Unicity tokens.
- **Trustless Execution:** The separation of the *Execute* and *Verify* functions allows agents to run on untrusted hardware. A host provides CPU cycles and storage, but the correctness of the computation is mathematically proven by the agent and verified by the recipient.

The Agentic Runtime Architecture

To enable this mobility, agents operate within a standardized **Agentic Runtime**. This runtime acts as an abstraction layer, sandwiching the agent between the raw infrastructure and the Unicity network.

1. **Executable Instance (The Agent):** Technically, an agent is a sandboxed computation (e.g., a Docker container or VM) that exposes two core procedures:

- `Execute(state, input)` $\texttt{new}_{state} : The logic defining state transitions.$ code Code

2. **Host-Independent Storage:** Agents do not rely on a host's local file system for persistence. Instead, the runtime interfaces with decentralized storage layers to pin and retrieve Unicity tokens. These tokens effectively act as the agent's hard drive, carrying its state history and assets in a portable, verifiable format.

3. **Overlay Transport Network:** Communication is handled via a circuit-switched overlay network. The runtime manages a Distributed Hash Table (DHT) or similar resolution service to map Agent IDs to physical entry points (e.g., URLs or relays). This allows agents to utilize any underlying transport protocol—from standard TCP/IP to short-wave radio in air-gapped environments—without altering the agent's internal logic.

4. **Secure Key Management:** Since hosts are generally untrusted, the runtime integrates with Hardware Security Modules (HSM) or Trusted Execution Environments (TEE). Cryptographic keys used for signing state transitions never leave the secure enclave, ensuring that the host operator cannot impersonate the agent.

AI-Powered Interoperability

A primary challenge in decentralized microservices is coordination: how does Agent A know how to format a request for Agent B without a rigid, pre-shared API specification? Unicity solves this through neuro-symbolic negotiation. Agents are equipped with Large Language Model (LLM) interfaces provided by the runtime. When two agents connect, they perform a semantic handshake:

- **Natural Language Discovery:** Agent A describes its intent in natural language (e.g., "I need to purchase storage space").
- **Dynamic Protocol Generation:** Agent B's LLM interprets the request, explains its capabilities, and the two agents negotiate a specific interface for that session.

- **Strict Execution:** Once the protocol is agreed upon, the agents revert to symbolic, deterministic code to execute the transaction and generate the Unicity proofs.

This allows for a "loose coupling" of agents. Developers do not need to build rigid adapters for every possible service provider; they simply instruct their agents on the goal, and the agents negotiate the technical implementation on the fly.

The Infrastructure Marketplace

The Agentic Internet creates a new peer-to-peer market for physical infrastructure. Individuals and organizations can participate as Infrastructure Providers, earning Unicity native currency by offering specific resources:

- **Compute Hosts:** Running agent runtimes (Docker/VM containers).
- **Relay Nodes:** Providing bandwidth and routing for the overlay network.
- **Bridge Agents:** acting as gateways between the Agentic Internet and the legacy Web2 world (e.g., serving a standard HTTP website backed by a swarm of decentralized agents).

This structure ensures that Unicity functions as a complete, self-sustaining ecosystem. It is not just a ledger for recording value, but a distributed computer where the logic of the machine economy lives, breathes, and transacts.