From Human-Centric to Agent-Native: Building Trustless Payment Infrastructure for Agentic AI

Kite AI Team

Link: Project Page | X Account

TL;DR: Kite enables AI agents to autonomously transact at scale with cryptographic safety and native x402 compatibility—solving the infrastructure crisis imprisoning the agent economy today.

While autonomous AI agents demonstrate production-grade multi-step reasoning capabilities today, **they remain constrained by the human-centric infrastructure**. Organizations face an impossible dilemma: grant agents financial authority and risk unbounded losses, or require manual authorization and eliminate autonomy. This infrastructure mismatch—not capability limitations—constitutes the primary bottleneck preventing realization of the projected $4.4 trillion agent economy.

This paper introduces Kite, the first comprehensive infrastructure system architected from first principles to treat AI agents as **first-class economic actors**. We address three fundamental failures of existing systems: **credential management complexity**, **payment infrastructure barriers**, and **unverifiable trust**. Our solution, the SPACE framework, along with **integratiblity with x402 protocol**, provides: (1) **Stablecoin-native payments** with sub-cent fees and instant finality; (2) **Programmable constraints** cryptographically enforced through smart contracts; (3) **Agent-first authentication** via hierarchical identity with mathematical delegation; (4) **Compliance-ready immutable audit trails**; and (5) **Economically viable micropayments** enabling pay-per-request pricing at global scale.

Kite's technical contributions include: (1) **A three-layer identity architecture** separating user (root authority), agent (delegated authority), and session (ephemeral authority) identities through BIP-32 hierarchical derivation; (2) **Programmable governance spanning services** through unified smart contract accounts where compositional rules enforce global constraints at the protocol level; (3) **Agent-native payment rails** using state channels that achieve sub-100ms latency at approximately $0.000001 per transaction. We provide **a formal security analysis** of our design. Our **multilayer revocation mechanism** combines immediate peer-to-peer propagation, cryptographic certificate verification, and economic slashing to ensure that compromised agents get terminated instantly.

Furthermore, Kite achieves **native compatibility with the x402 standard** alongside Google's A2A, Anthropic's MCP, OAuth 2.1, and the Agent Payment Protocol, positioning it as **a universal execution layer rather than an isolated protocol**. This level of interoperability enables seamless integration with existing ecosystems without bespoke adapters while delivering agent-native capabilities.

In summary, Kite provides the missing infrastructure layer that transforms autonomous agents from sophisticated chatbots into trustworthy economic actors—enabling the agent economy through mathematically guaranteed safety, not assumed trust, with x402 protocol for AI commerce at scale.

1. Entering the Agentic Future

1.1 The Promise and Prison of Autonomous AI

Artificial intelligence has reached an inflection point. GPT models process billions of daily requests, demonstrating remarkable accuracy on complex reasoning tasks. Claude orchestrates sophisticated multi-step workflows across thousands of function calls. Language models now reliably execute complex multi-step plans—the capability is proven and production-ready.

This technological maturity marks a fundamental shift from human-mediated interactions to agent-native autonomy. Today's autonomous agents analyze markets in milliseconds, optimize supply chains across continents, and execute investment strategies with precision that surpasses human capabilities. They coordinate distributed systems, manage portfolios worth billions, and make nuanced decisions at speeds and scales that would overwhelm entire teams of human operators. McKinsey projects these agents will generate $4.4 trillion in annual value by 2030, a conservative estimate given the exponential pace of adoption.

Yet a striking paradox emerges: the very infrastructure that should enable these agents instead imprisons them. **Autonomous agents remain severely constrained by infrastructure designed exclusively for human users.** An AI agent that analyzes global markets in microseconds must wait days for cross-border payment settlement. Agents trusted with critical business decisions cannot cryptographically prove they're operating within defined constraints. Users face the same impossible choice at the operational level: grant unconditional trust to agent monetary operations and risk everything, or manually authorize each transaction and destroy the very autonomy that makes agents valuable.

This isn't a capability problem. Infrastructure, not intelligence, has become the bottleneck. Three converging forces make an infrastructure revolution not just inevitable but urgent:

First, **model readiness**: Language models now reliably execute complex multi-step plans with production-grade consistency. The capability question is settled.

Second, **enterprise demand**: Companies face an impossible choice between granting agents real authority while risking catastrophic losses, or constraining them to advisory roles and forfeiting competitive advantage. The status quo is untenable.

Third, **regulatory reality**: Frameworks like the EU AI Act demand algorithmic accountability, requiring cryptographic proof of agent behavior. Compliance is no longer optional.

The models are ready. The businesses are desperate. The regulators are watching.

With Kite, the infrastructure revolution begins.

1.2 Why Existing Human-Centric Infrastructure Fails Autonomous Agents

1.2.1 Authentication vs Authorization: Twin Failures for Agent Systems

Contemporary digital services operate under a fundamental assumption: the entity requesting access is human. This assumption permeates every layer of security infrastructure, from username-password combinations to biometric verification systems. For AI agents, this creates distinct but interrelated failures in both authentication (verifying identity) and authorization (granting permissions).

Table 1: Human vs Agent Infrastructure Comparison

Dimension	**Human Infrastructure**	**Agent Reality**
Credential Management	Username/password, SSO	M×N API keys, no unified management
Identity Verification	Biometrics, 2FA, device fingerprinting	No cryptographic binding to principal
Authorization (OAuth)	One-time consent, session-based	Continuous operations, no runtime controls

The Credential Management Crisis Consider the operational reality of enterprise AI deployment. A single organization deploying 50 agents across 20 services must manage 1,000 unique credential relationships. Each relationship requires separate provisioning, rotation schedules, access policies, and audit trails. The complexity scales exponentially: M agents across N services generate M×N credential relationships, each with distinct security requirements and failure modes. This nightmare is already visible in today's Claude MCP integrations, where even a few deployments require managing dozens of API keys.

This proliferation creates cascading vulnerabilities. Long-lived API keys–service accounts, the current standard for agent authentication–possess dangerously broad permissions because narrowly-scoped credentials would require complex bidirectional configuration between every agent-service pair. When compromised, these keys grant attackers persistent access with permissions originally intended for humans or even administrators. The security model degrades to "security through obscurity," hoping credentials remain undiscovered rather than enforcing cryptographic constraints.

Identity Verification Impossibility The absence of cryptographic agent-to-principal binding creates a fundamental identity crisis. No service can definitively verify that "Alice's trading agent" genuinely belongs to Alice rather than a malicious actor claiming that relationship. Current authentication mechanisms cannot reliably distinguish between legitimate agents operating within defined parameters, compromised agents impersonating legitimate agents, or multiple agents falsely claiming the same principal.

This identity ambiguity forces services into binary trust decisions: grant full access based on credentials alone, or deny agent access entirely. Neither option enables the nuanced, constraint-based authorization that autonomous operations require.

OAuth's Blindness to Autonomous Behavior Traditional authorization frameworks like OAuth assume predictable, human-supervised API calls rather than autonomous agents executing thousands of non-deterministic operations per minute. While OAuth technically supports fine-grained scopes, the deeper problem is that these systems make authorization decisions once and then go blind. After an AI agent receives an OAuth token, the system cannot answer critical runtime questions such as whether the agent's behavior has suddenly changed in ways that suggest compromise, whether a given action will trigger a cascade of expensive operations across integrated services, or whether the cumulative cost of the agent's operations has exceeded reasonable bounds.

Current authorization models fail because they're:

- **Rigidly evaluated** - Permissions checked once at auth time, never reconsidered against changing context or costs
- **Economically blind** - No native tracking of resource consumption or spending velocity
- **Compositionally explosive** - Agents authorizing sub-agents create unpredictable permission chains impossible to audit

The result: each integrated service becomes a potential bankruptcy vector. A compromised agent with "payment processing" scope could execute thousands of micro-transactions below fraud thresholds. Even an agent with read-only access could perform costly DoS attacks by repeatedly querying expensive endpoints.

We need authorization systems that understand agent behavior, enforce economic boundaries, and make continuous trust decisions based on runtime context.

1.2.2 The Payment Infrastructure Mismatch

Traditional payment systems embody assumptions about transaction patterns that are antithetical to agent behavior. Where humans generate occasional, substantial transactions, agents produce continuous streams of micropayments. This fundamental mismatch creates economic and operational barriers that prevent agents from augmenting business and individual money movement workflows.

Economic Impossibility of Agent Transactions

The unit economics of traditional payments make agent operations financially prohibitive. Credit card processing incurs fixed costs ($0.30) plus percentage fees (2.9%), making small payments economically absurd. A penny transaction can cost over thirty cents to process—more than the actual payment itself. While local real-time payment systems like UPI, M-Pesa, and Alipay offer lower costs for human transactions, they too are fundamentally designed for person-to-person or person-to-merchant flows, not the high-frequency, API-driven micropayments that agents require. Agent operations naturally create many tiny transactions: an agent making thousands of API calls would rack up massive payment fees to transfer minimal actual value. This economic impossibility forces suboptimal workarounds that cripple agent effectiveness. Prepayment in large chunks locks up capital and creates reconciliation nightmares. Operation batching introduces dangerous latency in time-sensitive decisions. Delayed settlement leaves agents waiting for confirmation while opportunities vanish. Each compromise moves further from the real-time, pay-per-use model that agents require.

The global nature of AI agents collides with the regional reality of payments. International transfers cost $15-50 regardless of amount and impose multi-day delays. An agent coordinating suppliers across three continents faces payment friction that would paralyze any human operation, yet this is precisely the scale at which modern agents must operate.

Absence of Programmable Money

Traditional payments execute simple atomic transfers: move X from A to B. Agents require programmable money with complex conditional logic that current systems cannot express:

- **Streaming payments**: Continuous flows that adjust dynamically based on resource consumption

- **Conditional releases**: Funds released only upon cryptographic proof of completion
- **Automatic splits**: Proportional distribution calculated from real-time contribution metrics
- **Escrow with triggers**: Holdings released when oracle conditions verify
- **Recursive payments**: Transactions that programmatically spawn child transactions

Current systems require extensive external infrastructure to approximate these patterns, each layer introducing complexity, latency, and cascading points of failure. What should be simple programmable logic becomes a fragile tower of webhooks, databases, and reconciliation scripts.

1.2.3 The Trust and Programmability Void

The most fundamental limitation lies in the absence of verifiable, programmable trust. Current systems offer only binary trust models: complete access or complete denial. This void prevents agents from operating autonomously while remaining provably constrained within user-defined boundaries.

Unverifiable Constraint Enforcement

When users delegate authority to agents, they have no way to verify that the agents are actually respecting the constraints set for them. For instance, an investment agent might claim to enforce a 5% daily loss limit but could easily violate this rule without the user realizing until it is too late. Although agents function as black boxes by nature, current systems provide no cryptographic proof that the constraints were properly configured, that violations never occurred, or that the agent's reported actions faithfully reflect what actually happened.

Users discover breaches only after catastrophic damage—portfolios liquidated, accounts drained, data destroyed—with no verifiable record of what the agent actually did or when safeguards failed.

Programmable Governance Impossibility

Today, governance of agent behavior is fragmented across individual platforms, leaving users with no way to define or enforce unified policies. What's needed is **programmable governance**—rules that apply holistically across all services an agent touches. Instead of scattered per-platform controls, users should be able to encode global constraints like *"total spend across all agents under $1,000/day"* or *"any transaction above* $100 *requires user approval regardless of service."* Without such programmable, unified governance, agents can exploit gaps, distributing actions across platforms to bypass limits. With programmable governance, however, every transaction is checked against user-defined policies, ensuring agents operate within provable, system-wide guardrails.

Accountability and Recourse Vacuum

When agents misbehave, determining responsibility turns into manual forensic investigation rather than cryptographic verification. Without immutable audit trails, it becomes impossible to prove what actions the agent actually executed, whether those actions violated defined constraints, which component failed—be it model hallucination, platform error, or integration bug—or what the user originally authorized versus what ultimately occurred. While no system can guarantee an agent perfectly executes user intent (an impossibility given the non-deterministic nature of AI), the lack of cryptographic traceability means we cannot even establish what happened. Without this foundational

accountability layer—the ability to prove which agent did what, when, and under whose authority—resolution depends on lawyers instead of code. Users are left without automated kill switches to halt rogue agents, lack visibility into the chain of operations, and have no programmatic path to understanding failures, turning the promise of autonomous agents into a liability nightmare.

1.3 Stablecoin: The Native Currency for the Agentic Internet

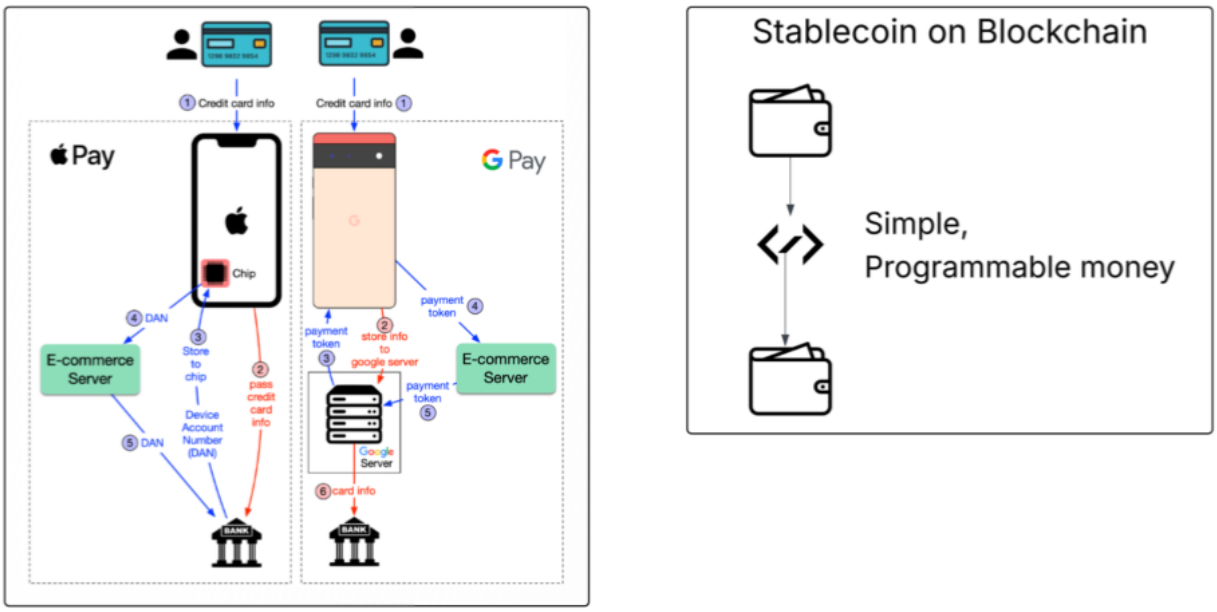

Fig. 1: Stablecoins as the Native Currency for the Agentic Internet

The solution begins with a radical reconceptualization of money itself. Digital versions of paper currency, while eliminating physical constraints, fail to address the fundamental economic and latency barriers facing agents. Agents require native currency that is machine-verifiable, programmable, and economical enough to meter each request, not monthly statements designed for human review, but streaming micropayments settled in real-time.

Traditional payment systems fail spectacularly in the agent context. A single credit card transaction traverses a complicated multi-party network: issuers, payment networks, tokenization services, acquirers, and banks. Each node adds latency, fees, and failure modes. All these result in high fixed fees, days-long settlement, and chargeback windows extending 120 days. These economics make agent-scale per-message pricing and automated machine-to-machine (M2M) micropayments impossible.

Blockchain-based stablecoins transform this equation entirely. A cryptographically signed stablecoin transfer from wallet to wallet achieves what traditional systems cannot:

- Transaction fees below one cent, enabling true pay-per-request economics
- Instant finality with zero chargeback risk

- Native programmability for conditional and streaming payments
- Global interoperability without cross-border friction

For these reasons, stablecoins aren't just an optimization. They provide programmable value transfer that operates at machine speed with machine precision, proving to be the fundamental primitive that makes the agentic economy possible.

1.4 The Incomplete Revolution: Why Existing Payments-First Blockchain Isn't Enough

Recent developments in blockchain architecture have addressed a critical insight: payments and general computation require fundamentally different optimizations. While Ethereum pioneered programmable money, its design optimizes for maximum expressiveness rather than payment efficiency. This creates a paradox where the most sophisticated smart contract platform struggles with simple payment operations due to high costs, unpredictable fees, and congested block space during peak demand.

Tempo, introduced by Paradigm, represents a pivotal shift in blockchain design philosophy, a "payments-first" architecture that prioritizes transaction efficiency over computational generality. This approach acknowledges that 90% of blockchain activity involves simple value transfers, not complex smart contracts. By optimizing for this reality, payments-first chains achieve dramatic improvements in cost, speed, and predictability.

Tempo's architecture delivers three critical optimizations:

- **Predictable stablecoin fees**: Eliminates gas token volatility by charging fees directly in stablecoins
- **Opt-in privacy**: Transactions remain private by default, with cryptographic proofs enabling compliance when required
- **Dedicated payment lanes**: Isolated blockspace for transfers, preventing congestion from other activities

Yet even Tempo's payments-first design lacks the agent-native infrastructure required for autonomous operations:

- **No wallet hierarchy**: Without separate tiers for agents and principals, security boundaries blur dangerously
- **Missing programmable guardrails**: Agents will hallucinate and make errors—only cryptographic spending rules can prevent catastrophic losses
- **Absent agent transaction types**: No native support for embedding API requests within payments
- **Zero protocol compatibility**: Lacks integration with essential agent standards like A2A, MCP, AP2, and OAuth 2.1

Without these agent-specific layers, organizations cannot confidently delegate payment authority. The vision of transforming every agent interaction into a compliant, auditable micropayment remains unrealized.

Tempo solves the payment problem but not the agent problem. The promise of autonomous agents remains unfulfilled, and thus the agentic infrastructure revolution is still incomplete.

1.5 The SPACE Framework: End-to-End Infrastructure Reimagined for Agents

This paper introduces Kite, the first infrastructure system designed from first principles for the agentic economy. We present the **SPACE** framework as the complete solution:

- **S**tablecoin-native: Every transaction settles in stablecoins with predictable sub-cent fees
- **P**rogrammable constraints: Spending rules enforced cryptographically, not through trust
- **A**gent-first authentication: Hierarchical wallets with cryptographic principal binding
- **C**ompliance-ready: Immutable audit trails with privacy-preserving selective disclosure
- **E**conomically viable micropayments: True pay-per-request economics at global scale

Our Core Innovations The core innovations of Kite include:

Three-Layer Identity Architecture
We introduce the first hierarchical identity model that separates user (root authority), agent (delegated authority), and session (ephemeral authority) identities. Each agent receives its own deterministic address derived from the user's wallet using BIP-32, while session keys are completely random and expire after use. Sessions are authorized by their parent agent through cryptographic signatures, creating a clear delegation chain from user to agent to session. This defense-in-depth architecture ensures graduated security: compromising a session affects only one delegation; compromising an agent remains bounded by user-imposed constraints; and user keys secured in local enclaves inaccessible to any external third parties, which are thus highly unlikely to be compromised, represent the only point of potential unbounded loss. While funds remain compartmentalized for security, reputation flows globally across the system. Every transaction and interaction contributes to a unified reputation score, establishing trust that spans users, agents, and services throughout the Kite platform.

Programmable Governance Beyond Smart Contracts
While smart contracts enable programmable money, agents require compositional rules that span multiple services. Kite implements a unified smart contract account model where users own a single on-chain account holding shared funds. Multiple verified agents operate through session keys with cryptographically enforced spending rules: "Expense agent limit $5,000/month for reimbursements, Scheduling agent limit $500/month for bookings, Grocery agent limit $2,000/month with velocity controls, Investment agent limit $1,000/week with volatility triggers." Rules can be temporal (e.g., increase limits over time), conditional (e.g., reduce limits if volatility spikes), and hierarchical (cascade through delegation levels). These aren't policies, but programmatically enforced boundaries.

Agent-Native Payment Rails with State Channels
Beyond stablecoins and payments-first blockchains, the real revolution goes deeper. Kite creates agent-first transaction types. Beyond simple transfers, Kite implements programmable micropayment channels optimized for agent patterns. Instead of the traditional over-complicated multi-party card rails of authenticate, request, pay, wait, and verify, payments are instantly settled during agent interaction within the same channel. Two on-chain transactions (open and close) enable thousands of off-chain signed updates, achieving sub-hundred-millisecond latency at $1 per million requests. This

architectural inversion, treating per-request and streaming micropayments as first-class behaviors with instant finality, unlocks agent-native economics which were previously impossible.

In the following chapters, we will detail how Kite's architecture, protocols, and implementations solve each infrastructure failure, transforming the promise of the agentic economy into operational reality.

The agentic future isn't waiting for better models. It's waiting for infrastructure that treats agents as first-class economic actors.

With Kite, the wait ends.

2. Kite AI Design Principles

2.1 Tailored Design for the Future of Agentic AI

The evolution from human infrastructure to agent infrastructure requires fundamental architectural reimagination. Every existing blockchain, from Bitcoin to Ethereum to modern payments chains like Tempo, embeds a critical assumption: transactions are initiated by humans who can manage private keys, evaluate risks in real time, and intervene when necessary. This assumption shapes everything from authentication to settlement timing.

Kite breaks this assumption entirely. Built from first principles to treat AI agents as first-class citizens in the digital economy, each agent maintains its own cryptographic identity, authentication mechanisms, wallet hierarchy, and governance policies. This creates a system where agents authenticate, transact, and coordinate natively without human workarounds.

The implications cascade through every layer. Manual key management becomes hierarchical key derivation. Transaction evaluation becomes programmable constraint enforcement. Social reputation becomes cryptographic verification. These inversions of design assumptions transform not just how agents operate, but what becomes possible when infrastructure aligns with agent capabilities rather than fighting them.

Consider the difference in practice. A human evaluates a $1,000 transaction by reading the amount, recognizing the recipient, and clicking confirm. An agent executing the same transaction must prove its authorization cryptographically, demonstrate the transaction falls within programmed boundaries, generate an immutable audit trail, and settle instantly to continue its workflow. The human process takes seconds of attention. The agent process takes milliseconds of computation. Yet current infrastructure forces agents through the human process, destroying their advantages.

2.2 Chain of Trust with Cryptographic Proof

Every action in the Kite system creates a cryptographically verifiable audit trail, establishing an immutable chain of custody from user to agent to service to outcome. This transforms agent operations from black-box mysteries into transparent, provable sequences where every decision can be verified and every constraint can be enforced mathematically.

The chain of trust operates through three critical principles:

No Direct Key Access: The revolution starts with a simple insight: agents should never touch private keys directly. Instead, each operation receives a one-time, task-scoped session key with surgical

precision permissions. An agent authorized to purchase data feeds receives a key valid only for specific providers, exact amounts, and narrow time windows. When the task completes or the window expires, the key becomes cryptographically useless. Even total compromise affects only that single operation.

Fine-Grained Task Authorization: Permissions are scoped at the task level, not the agent level. Both users and agents can create these scoped authorizations—users delegate to agents, and agents can further sub-delegate to specialized sub-agents or tasks. This granular authorization extends beyond simple spending limits. Permissions scope down to individual API endpoints, specific data types, and conditional triggers. An agent authorized to "purchase data feeds" receives a session key valid only for strictly enforced specific data providers, amounts, and time windows. This granularity makes broad compromises mathematically impossible.

Reputation Without Identity Leakage: The relationship between users and agents creates a fascinating paradox: shared reputation with independent identity. Each user and corresponding agent accumulates their own track record of successful operations, building trust through performance. For each user-agent binding, the reputation of the duo is inherited as a combination of the user's reputation and the agent's reputation, and the operation outcome affects both the reputation of the user and the agent, where the cryptographic binding to the controlling user ensures accountability flows upward. Services know that behind every agent stands a real user with real stakes, even when that user's identity remains private. Users can selectively disclose ownership when beneficial, but privacy remains the default.

This architecture enables something unprecedented: complete transparency without sacrificing privacy. Regulators can verify that agents operated within legal boundaries. Services can confirm payment authorization. Users can audit every action their agents took. Yet sensitive business logic, trading strategies, and personal information remain encrypted. The system proves compliance without revealing secrets.

2.3 Sovereignty through Separation

Kite rigorously separates decentralized asset management from developer services, creating an architecture where security and usability reinforce rather than compromise each other.

Decentralized Asset Model: The decentralized asset model ensures complete user sovereignty. All funds remain in self-custodial wallets controlled by smart contracts that even Kite cannot access. Users maintain exclusive control over their digital wealth while agents operate within mathematically enforced boundaries. Users maintain independent access to their funds through standard blockchain interfaces, regardless of Kite's availability. This isn't a promise or a policy. It's a cryptographic guarantee.

Kite Platform Services: Meanwhile, the Kite platform provides the abstraction layer that makes agent operations practical. APIs handle complex cryptographic operations, protocol translations, and cross-chain interactions. Developers work with familiar patterns while the platform manages key derivation, session management, and constraint compilation. The platform never touches assets directly. It simply provides the tools to make asset operations seamless.

This separation solves a fundamental tension in blockchain systems. Pure decentralization often means terrible user experience. Pure centralization means unacceptable risk of single point failure and user

assets compromise. Kite achieves both security and usability by separating concerns architecturally. Assets remain decentralized and sovereign. Services remain centralized and optimized. Users get the best of both worlds without compromise.

The implications extend beyond technical architecture. Regulators can audit the platform without accessing user funds. Developers can build sophisticated applications without managing private keys. Users can delegate complex operations without surrendering control. Each stakeholder operates in their domain of expertise while the system maintains security guarantees across all layers.

2.4 Native Compatibility: Zero Friction to Integrate With Existing Standards

Rather than creating another isolated protocol that fragments the ecosystem further, Kite embraces existing standards as first principles. This isn't grudging compatibility or minimal compliance. It's architectural integration that makes Kite agents native citizens of multiple protocols simultaneously.

x402 standard defines a common, agent-first payment flow and message schema so any compliant service can accept payments from any compliant agent without bespoke adapters. In Kite, x402 serves as the interoperability layer between agents and services: agents convey payment intents; services verify authorization and terms; settlement details travel in a standard, machine-actionable envelope. The **Agent Payment Protocol (APP)** then specializes this flow for stablecoin settlement on Kite, preserving broad compatibility while providing a practical default path today.

Google's A2A protocol enables direct agent coordination across platforms. Kite agents speak A2A fluently, coordinating with agents from any ecosystem using established communication patterns. When a Kite agent needs to coordinate with a Google agent to complete a complex workflow, they communicate directly without translation layers or compatibility bridges.

Google's Agent Payment Protocol is a neutral, open protocol that defines how agent payments should be expressed, it is like ERC20. Kite plays the Ethereum role: the execution + settlement layer for those AP2 intents. A Kite agent takes an AP2-compliant mandate and enforces it on-chain with programmable spend rules, gasless/bundled ops, and stablecoin-native settlement.

Anthropic's MCP ensures model interoperability across the entire LLM ecosystem. Claude, GPT, and emerging models interact through the same protocol layer, making model choice a business decision rather than a technical constraint. A Kite agent can leverage Claude for reasoning, GPT for generation, and specialized models for domain tasks, all within a single workflow.

OAuth 2.1 compatibility means compatibility with human-centric services and existing login/payment flows, thus existing services don't need to rebuild their authentication systems. A service supporting OAuth today can accept Kite agents tomorrow with minimal changes. This backward compatibility creates a gradual migration path for developers where they can adopt Kite AI infrastructure without abandoning the broader internet and enterprise ecosystems. Together with x402 standard compatibility, agent-native payment flows are connected with today's service ecosystems without demanding bespoke adapters or rewrites.

These standards embrace transforming adoption dynamics. Developers don't choose between Kite and their existing stack. They add Kite to enhance what already works. Services don't rebuild for agents. They extend existing APIs with agent awareness.

The future arrives through evolution, not revolution.

3. Architecture

3.1 Overview

Kite's architecture represents a purposeful departure from traditional blockchain design. Instead of building another general purpose chain that happens to support payments, every architectural decision optimizes for one goal: enabling autonomous agents to operate with mathematical safety guarantees.

The architecture stacks purpose-built layers from blockchain to applications, each solving specific challenges in the agent autonomy puzzle:

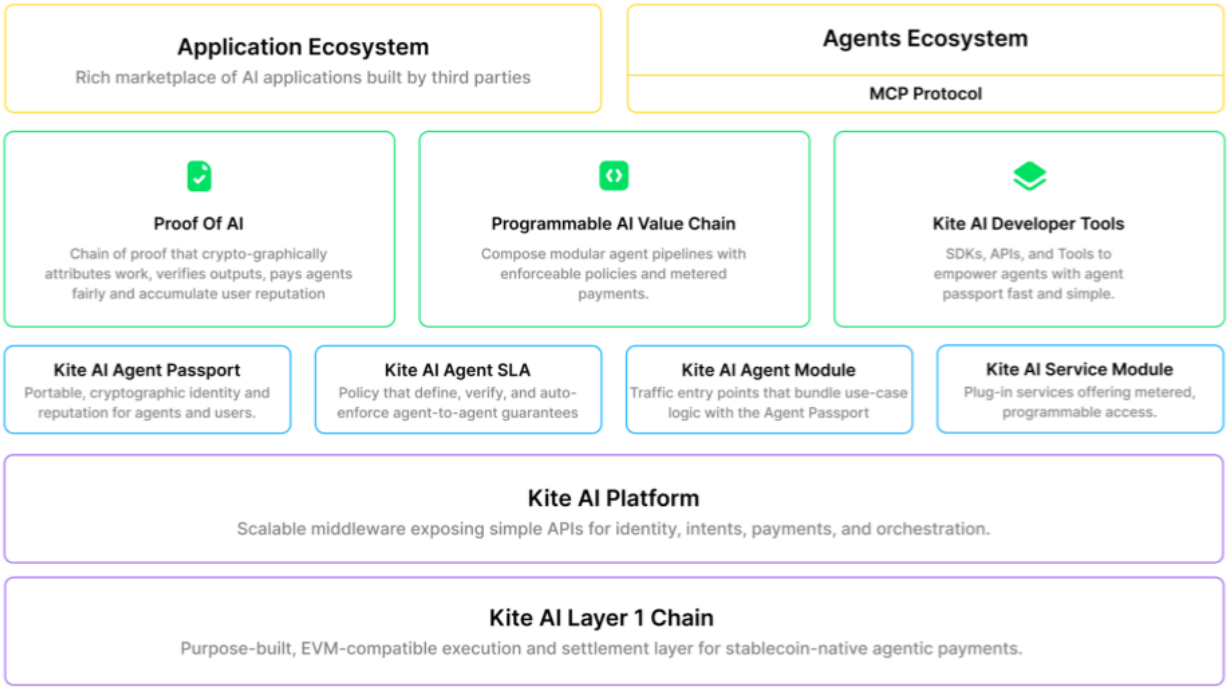

Fig. 2: Overview of Kite's Purpose-built Layered Architecture

Base Layer: An EVM-compatible L1 optimized for stablecoin payments, state channels, and settlement. Unlike general-purpose chains, every optimization targets agent transaction patterns.

Platform Layer: Abstracts blockchain complexity through agent-ready APIs for identity, authorization, payments, and SLA enforcement. Developers interact with familiar patterns while the platform handles cryptographic proofs and on-chain settlement.

Programmable Trust Layer: Introduces novel primitives including Kite Passport (cryptographic agent IDs), Agent SLAs (smart contract interaction templates), and compatibility bridges to A2A, MCP, and OAuth 2.1. This layer ensures agents can interact with both Web3 and traditional services seamlessly.

Ecosystem Layer: Two interconnected marketplaces—an Application Marketplace for AI services and an Agents Ecosystem connected through standard protocols. Services register once and become discoverable by millions of agents.

Together, these layers let agents transact with predictable fees, enforce SLAs, and accumulate reputation across accounts and interactions. Kite architecture achieves transformative results: dramatic reduction in payment costs through state channels, significant latency improvement via dedicated agentic payment lanes, and complete elimination of credential management overhead through cryptographic identity. More critically, true agent autonomy is realized with programmable guarantee of constraint compliance.

3.2 Terminology and Core Concepts

Understanding Kite requires grasping the fundamental building blocks that enable autonomous agent operations. These aren't merely technical definitions but the architectural choices that transform AI agents from sophisticated chatbots into economic actors.

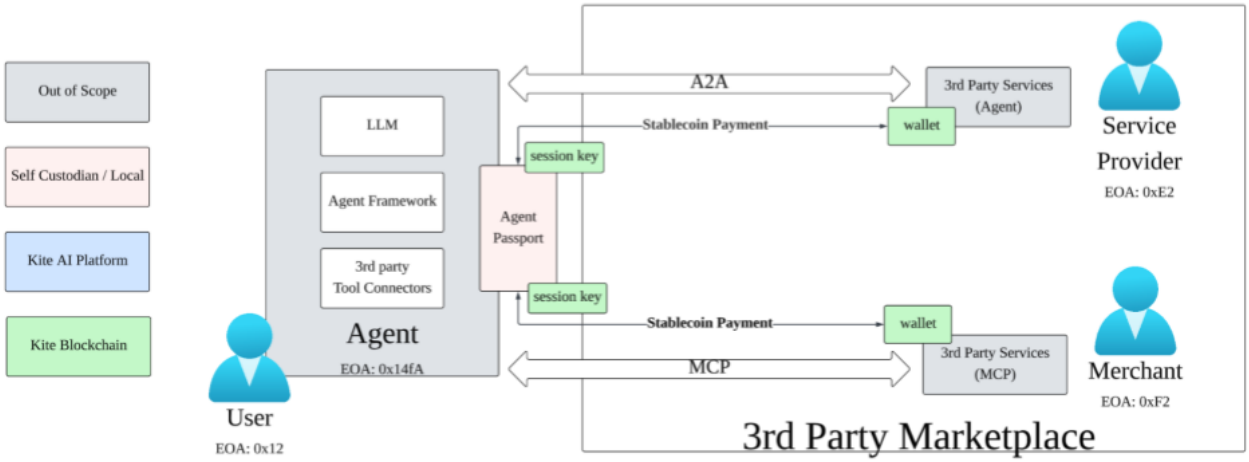

Fig. 3: Illustration of Kite AI Core Entities and Workflow

3.2.1 Core Entities

The Kite ecosystem operates through four fundamental entity types, each playing a critical role in the agent economy.

User: The human principal who owns and controls a fleet of AI agents. More than account holders, users represent the bridge between traditional legal frameworks and autonomous systems. They delegate specific capabilities to agents while maintaining ultimate authority, manage master wallets that serve as the root of cryptographic trust, and set global policies that cascade through all their agents. Users remain the legally responsible entities, ensuring that agent autonomy never means absence of accountability.

Agent:A personal AI assistant or autonomous program acting on behalf of a user. These are sophisticated entities that execute complex tasks, interact with multiple services simultaneously, and handle money within cryptographically enforced boundaries. Each agent maintains its own wallet, accumulates its own reputation, and operates under its own governance policies. Yet they remain mathematically bound to their controlling user through BIP-32 derivation, creating provable ownership without key exposure.

Service: Any external offering that agents interact with to accomplish tasks. Services span from data APIs providing real-time market feeds to GPU providers offering inference compute, from SaaS applications exposing business logic to other agents offering specialized capabilities. Services can integrate through multiple protocols: MCP for model interactions, A2A for agent coordination, or traditional OAuth for legacy compatibility. Each service maintains sovereignty over its access policies while participating in the global agent marketplace.

Merchant/Provider: The businesses, developers, or infrastructure operators who make services discoverable and consumable by millions of agents. Providers don't just list APIs; they define SLAs with automatic penalties, establish reputation through verifiable performance, and participate in programmable commerce where every interaction becomes a micropayment. The merchant role transforms B2B services from manual integration nightmares into plug-and-play agent resources.

3.2.2 Identity and Trust Infrastructure

Identity in the agent economy transcends usernames and passwords. It requires cryptographic proof, verifiable delegation, and portable reputation.

Kite Passport: The cryptographic identity card that makes agent operations possible. Unlike traditional credentials that merely authenticate, the passport creates a complete trust chain from user to agent to action. It binds to existing identities like Gmail or Twitter through cryptographic proofs, enabling users to leverage their existing digital presence. The passport contains not just identity but capabilities: what an agent can do, how much it can spend, which services it can access. Most critically, it enables selective disclosure. An agent can prove it belongs to a verified human without revealing which human, preserving privacy while maintaining accountability.

Decentralized Identifier (DID): A globally unique, cryptographically verifiable identifier that establishes immutable binding between agents and users. DIDs aren't random strings but structured identifiers that encode relationships. A user might have did:kite:alice.eth while her trading agent has did:kite:alice.eth/chatgpt/portfolio-manager-v1. This hierarchy makes authority chains instantly verifiable. Any service can cryptographically confirm that a session belongs to an agent, that agent belongs to a user, and that user authorized the current operation. No central authority needed, no API calls required, just mathematical proof.

Verifiable Credentials (VCs): Cryptographic attestations that prove specific capabilities or authorizations. A VC might certify that an agent passed compliance training, holds a trading license, or maintains a reputation above a threshold. These aren't self-declared claims but cryptographically signed attestations from trusted authorities (e.g., educational email verification). VCs enable fine-grained access control where services can require specific proofs before granting access. An exchange might require KYC verification, a data provider might require payment history, a compute cluster might require reputation scores. The beauty lies in composability: agents accumulate credentials over time, building portable proof of capabilities.

Proof of AI: Every agent action creates an immutable, tamper-evident log anchored to the blockchain. These aren't just logs but cryptographic proof chains of what occurred. They establish complete lineage from user authorization through agent decision to final outcome. When disputes arise, proof chains provide indisputable evidence. When regulators investigate, they see complete transparency. When users audit their agents, they verify every action. Traditional systems rely on logs that can be altered or deleted. Proof chains provide mathematical certainty about historical events.

3.2.3 Wallets Structure

The agent economy demands sophisticated wallet structures that enable programmable control without sacrificing security.

EOA Wallet (Externally Owned Account): The traditional blockchain wallet controlled by a private key. In Kite's architecture, users maintain master EOA wallets that serve as the root of authority. These wallets live in secure enclaves, hardware security modules, or user devices, never exposed to agents or services. The EOA signs the initial authorizations that delegate specific powers to agent operations, creating the foundation of the trust chain. While simple in concept, EOAs become powerful when combined with smart contract delegation.

AA Wallet (Smart Contract Account): The revolutionary advance that makes agent payments possible. Account Abstraction wallets aren't just addresses but programmable accounts with built-in logic. AA enables developers to write smart contract code that governs spending, bundles transaction execution, and allows third parties to pay gas fees. Unlike traditional account models, Kite's AA implementation enables multiple agents to operate from a single treasury with cryptographically isolated permissions. In Kite's model, the user owns a single on-chain AA account holding shared funds in stablecoins. Multiple agents operate this account via session keys, but only within their authorized limits. One treasury, multiple operators, perfect isolation.

Embedded Wallets: Self-custodial wallets integrated directly into applications, abstracting complexity while maintaining sovereignty. Users don't manage seed phrases or private keys, yet they maintain complete control over funds. Embedded wallets enable one-click agent authorization, automatic session management, and transparent fund flows. They make blockchain invisible to users who think in their local currency, not tokens, while preserving the cryptographic guarantees that make agent operations safe.

3.2.4 Payment Architecture

Kite's payment system consists of two coordinated components: (i) the on-chain Agent Payment Protocol, which enforces programmable flows and policy limits; and (ii) the On/Off-Ramp API, which mediates fiat and stablecoin ingress/egress. Users pre-fund agent wallets, agents spend under explicit authorization, and merchants settle in their preferred currency. The components are detailed below.

Agent Payment Protocol: A comprehensive system that enables payment flows impossible with traditional infrastructure. This protocol supports micropayments down to fractions of cents, streaming payments that flow continuously based on usage, pay-per-inference models where every API call carries value, and conditional payments that release based on performance. The Agent Payment Protocol enables instant, global, programmable value transfer at machine speed.

On/Off-Ramp API: The bridge between traditional finance and the agent economy. Through integration with global providers like Coinbase, PayPal, users fund agent wallets with local payment methods, stablecoins, and bank accounts while merchants withdraw earnings to bank accounts, leveraging Kite's identity and trust infrastructure. The ramp handles compliance, fraud prevention, and currency conversion invisibly. Users never need to understand blockchain; they just see their local currency in and out. This abstraction layer makes agent payments accessible to billions who will never own crypto but need agent services.

3.2.5 Governance and Safety Mechanisms

Autonomous agents require sophisticated governance that ensures safety without sacrificing capability.

SLA (Service-Level Agreement) Contracts: Smart contracts that transform vague service promises into mathematically enforced guarantees. Unlike traditional SLAs that rely on legal enforcement, these contracts automatically execute penalties and rewards. An SLA might specify 99.9% uptime with automatic pro-rata refunds for downtime, response times under 100ms with tiered pricing based on performance, or data accuracy requirements with slashing for errors. These programmable agreements create trust through code rather than courts.

Programmable Trust/Intent-Based Authorization: Users express their intentions through mathematical constraints that compile to blockchain enforcement. Instead of hoping agents respect policies, the system ensures they cannot violate them. Intents can specify spending caps that cannot be exceeded even if the agent tries, temporal windows outside which operations automatically fail, whitelisted merchants or blacklisted categories enforced at protocol level, and complex conditional logic like "if volatility exceeds 20%, reduce limits by half." These intents automatically expire, preventing forgotten authorizations from becoming vulnerabilities. The user's intent becomes immutable law.

Session Keys/Ephemeral Keys: Temporary cryptographic keys that implement zero-trust session management. Generated for each agent task, these keys are completely random, never derived from permanent keys, ensuring perfect forward secrecy. A session key might authorize "transfer maximum $10 to providers A, B, or C for data feeds between 2:00 PM and 2:05 PM today." The key executes its authorized operation then becomes cryptographically void forever. Even total session compromise affects only one operation for minutes with bounded value. This architecture prevents the cascading failures that plague traditional API key systems where one breach means total compromise.

Reputation System: Trust scores that accumulate based on verifiable behavior rather than self-reported ratings. Every successful payment increases reputation. Every failed delivery decreases it. Every policy violation triggers penalties. But unlike traditional ratings that can be gamed, Kite's reputation derives from cryptographic proofs of actual behavior. High reputation agents access better rates, higher limits, and premium services. Low reputation agents face restrictions and additional verification. Reputation becomes portable across services, solving the cold-start problem where new relationships begin from zero trust. An agent with proven history on one platform can present verifiable credentials to new services, bootstrapping trust through cryptographic proof rather than promises.

3.3 Detailed Design

3.3.1 The Three Layer Identity Imperative

Human payment systems recognize only two entities: the payer and the payee. This binary model suffices when humans initiate every transaction, evaluate every risk, and maintain control over every key. Agent systems shatter this assumption. They require a third layer that enables delegation without the catastrophic vulnerability of key sharing.

User Identity (Root Authority)
The user maintains ultimate control as the cryptographic root of trust. Their private keys live in secure enclaves, hardware security modules, or protected device storage, never exposed to agents, services, or even the Kite platform itself. Users can instantly revoke all delegated permissions with a single transaction, set global constraints that cascade through all agents, and monitor every operation through immutable proof chains. This isn't theoretical control through terms of service; it's mathematical control through cryptographic enforcement.

Agent Identity (Delegated Authority)
Each AI agent receives its own deterministic address mathematically derived from the user's wallet using BIP-32 hierarchical key derivation. When you create a ChatGPT agent for portfolio management, it gets address 0x891h42Kk9634C0532925a3b844Bc9e7595f0eB8C, provably linked to your wallet yet cryptographically isolated. This address serves as the agent's on chain identity, enabling it to authorize sessions for spending operations.

The mathematical derivation creates powerful properties. Anyone can verify the agent belongs to you through cryptographic proof, yet the agent cannot reverse the derivation to access your private key. The agent maintains its own reputation score, coordinates with other agents autonomously, and operates within user defined boundaries that the blockchain enforces absolutely. Even total agent compromise remains bounded by smart contract constraints.

Session Identity (Ephemeral Authority)
For each task execution, the system generates a completely random session key with address 0x333n88Pq5544D0643036b4c955Cc8f8706g1dD9E. These keys are never derived from wallet or agent keys, ensuring perfect forward secrecy. They're single-use authorization tokens that execute specific actions without exposing permanent credentials.

The session key generation happens entirely locally, without server communication or private key transmission. Once generated, a DID session registers in the agent network, self-containing the proof of authority, chain of trust, and validity period. The session validates through its time window then becomes permanently invalid. Even quantum computers cannot resurrect expired sessions.

This three tier model provides defense in depth that payments first chains lack. Compromising a session affects only one transaction. Compromising an agent is limited by user imposed constraints. Only user key compromise enables unbounded loss, and secure enclave protection makes this nearly impossible.

Identity Resolution Through Standards

Kite extends ENS standards to claim agent ownership with human readable identifiers:

- User ID: did:kite:alice.eth

- Agent ID: did:kite:alice.eth/chatgpt/portfolio-manager-v1

Public resolvers enable instant verification:

- GetAgent(AgentID) → AgentID, AgentDomain, AgentAddress
- ResolveAgentByDomain(AgentDomain) → AgentID, AgentDomain, AgentAddress
- ResolveAgentByAddress(AgentAddress) → AgentID, AgentDomain, AgentAddress
- GetAgentBySession(SessionID) → AgentID, AgentDomain, AgentAddress, SessionInfo

Any service can verify the complete authority chain without contacting Kite or the user, enabling permissionless interoperability.

3.3.2 Programmable Governance and Money

Smart contracts revolutionized programmable money, but agents require programmable governance that spans services, evolves over time, and responds to changing conditions. These aren't simple spending limits; they're sophisticated control systems that make agent autonomy safe.

Compositional Rules Across Services
Constraints combine through boolean logic to create sophisticated policies. "Total spending across all platforms less than $1000/day AND no single transaction greater than $100 AND only verified providers." These rules compile to smart contract code that evaluates atomically. Agents cannot circumvent limits by splitting transactions or distributing operations across services. The blockchain becomes the ultimate arbiter, enforcing rules with mathematical certainty.

Temporal Evolution of Trust
Static limits ignore the reality that relationships build over time. Kite implements progressive trust: "Start with $10/day limit, increase by $10 weekly up to $100/day after trust is established." The blockchain automatically adjusts limits based on time and behavior. No manual intervention, no forgotten updates, just programmatic trust evolution that reflects actual agent performance.

Conditional Response to External Signals
Markets change, threats emerge, opportunities appear. Agent constraints must adapt: "If volatility is greater than 20%, reduce trading limit by 50%." Oracle networks feed real time signals into smart contracts, triggering automatic adjustments. When markets panic, agents automatically become more conservative. When security breaches occur, new authorizations freeze. The system responds to threats faster than humans can react.

Hierarchical Cascade Through Organizations
Enterprise deployments require nested governance: "ChatGPT agent limit $10,000/month, Cursor limit $2,000/month, other agents limit $500/month." Constraints cascade through delegation levels, with child limits automatically bounded by parent limits. A department cannot exceed its division's budget. An agent cannot exceed its tier's allocation. Organizational policies propagate cryptographically.

The Unified Account Model

Kite designs a unified smart contract account model that elegantly balances simplicity with control. The user (EOA 0xUser) owns a single on-chain account holding shared funds in USDC or pyUSD.

Multiple verified agents—Claude, ChatGPT, Cursor—each operate this account via their own session keys (0xSession01 through 03), but only through sessions that enforce rules and quotas.

When an agent executes a task, spending comes from the shared pool within its session allowance, and the account pays merchants programmatically. Result: one treasury, per session risk isolation, and fine grained, auditable control across all the user's agents. No fund fragmentation, no complex reconciliation, just elegant unified management.

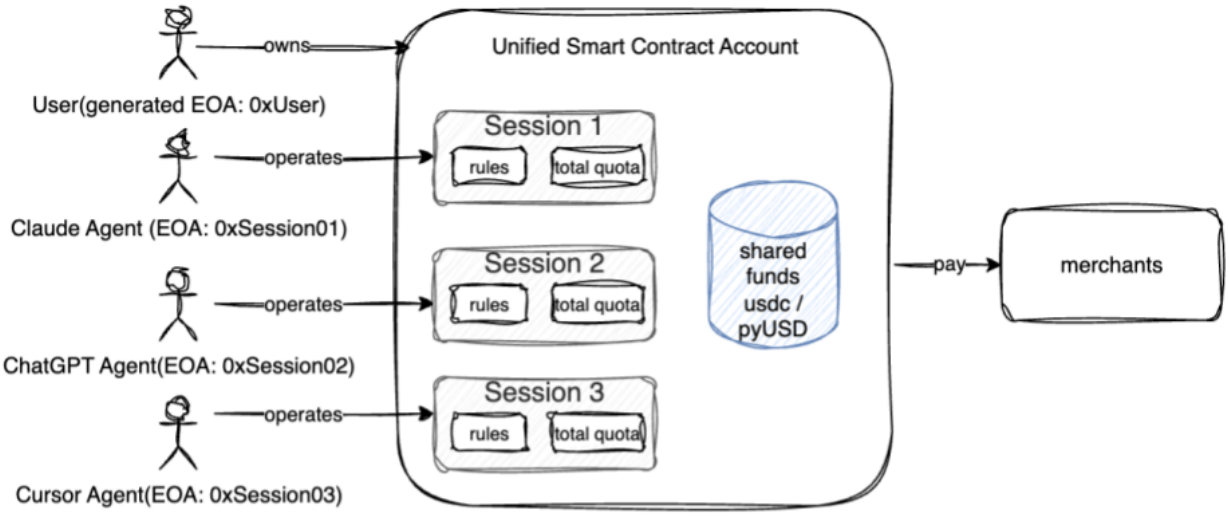

Fig. 4: The Unified Account Model

Spending Rules versus Policy

The system distinguishes between on chain rules and off chain policies based on their evaluation requirements:

Dimension	Spending Rules	Policy
Scope	Asset or stablecoin related, leveraging programmable money	Full control and flexibility for complex logic
Evaluation	Entirely on chain through smart contracts, ensuring transparency	Securely off chain in user's local or Kite's TEE
Interoperability	Integrates with any on chain account, fully decentralized	Platform specific but can integrate third party systems
Use Cases	Spending limits, rolling windows	Session TTL, categories, recipient lists

Session Key Implementation with Smart Contracts

The AA wallet smart contract implements session keys through a plugin architecture:

```
addSessionKeyRule(
    address sessionKeyAddress,  // The newly generated session key
    bytes32 agentId,           // Agent performing the actions
    bytes4 functionSelector,    // Allowed function (e.g., pay())
    uint256 valueLimit          // Maximum transaction value
)
```

The user signs authorization transactions with their EOA wallet, representing cryptographic intent. The transaction packages as a UserOperation and sends to a bundler for on chain execution. Sessions expire automatically when validUntil timestamps pass, or users can actively revoke by calling removeSessionKeyPermission().

Programmable Escrow Contracts

Kite extends beyond direct payments by introducing programmable escrow mechanisms that embed smart contracts between agents and merchants. In this framework, buyers issue cryptographically signed payment intents, authorizing funds into escrow accounts with defined expiry conditions. These escrows can be partially or fully released based on verified outcomes, supporting the complete lifecycle of commercial interactions including authorization, capture, charge, void, reclaim, and refund. The design remains noncustodial and permissionless, ensuring that users retain full control over assets throughout the process.

The protocol supports the inclusion of additional operators such as third-party entities that can relay transactions or cover gas costs while remaining cryptographically constrained by the payer's signed hash. Leveraging standards such as ERC 3009, Kite enables signature based gasless preapprovals that further streamline execution and user experience.

As the escrow layer is implemented as code, it can be extended with programmable business logic including dynamic revenue sharing, token swaps, conditional privacy modules, or integration with other smart contracts. Each transaction thereby becomes an auditable, reversible, and composable financial primitive, aligning with Kite's goal of transforming payments into transparent, verifiable, and programmable economic interactions.

3.3.3 Compatible with Agent Protocols

Modern agents don't operate in isolation. They integrate with Google's A2A protocol, Anthropic's MCP, and enterprise OAuth systems. Payment infrastructure must provide native bridges to these ecosystems, not awkward adapters.

Protocol Translation
Seamless mapping between agent communication standards enables universal interoperability. A Kite agent speaks A2A to Google agents, MCP to Claude, OAuth to enterprise services—all through the same identity and payment rails. This isn't middleware translation; it's native multilingual capability.

Identity Federation
Unified authentication eliminates the M×N credential problem. Users authenticate once with existing providers; agents inherit permissions across all connected services. No duplicate credentials, no synchronization nightmares, just elegant identity flow.

Compliance Frameworks
Built in support for regulatory requirements across jurisdictions ensures global operability. GDPR in Europe, CCPA in California, data residency requirements in China—all handled automatically through configurable compliance modules.

The integration happens through extended protocol definitions:

```
// AgentDID definition
message AgentDID {
 string did = 1;                  // Unique identifier for the agent
 string user_wallet = 2;          // User's wallet address who controls this agent

 repeated VerificationMethod verification_methods = 3;
 AgentProvider provider = 4;

 int64 created = 5;               // Timestamp of DID creation
 int64 updated = 6;               // Last update timestamp
}

message VerificationMethod {
 string id = 1;                   // Verification key identifier
 string type = 2;                 // Always 'EcdsaSecp256k1VerificationKey2019'
 string controller = 3;           // DID that controls this key
 string public_key_hex = 4;       // Agent's public key for verification
}
```

Combined with the A2A protocol:

```
// AgentCard conveys key information: Key information uses a bold and larger fond.
message AgentCard {
// New DID we proposed.
 AgentDID id = 19;
 // The service provider of the agent, like ChatGPT
 AgentProvider provider = 4;
 . . . . . .
 // description of agents
 // JSON Web Signatures computed for this AgentCard.
 repeated AgentCardSignature signatures = 17;
}

// AgentCardSignature represents a JWS signature of an AgentCard.
// This follows the JSON format of an RFC 7515 JSON Web Signature (JWS).
message AgentCardSignature {
 // The protected JWS header for the signature. This is always a
 // base64url-encoded JSON object. Required.
 string protected = 1 [(google.api.field_behavior) = REQUIRED];
 // The computed signature, base64url-encoded. Required.
 string signature = 2 [(google.api.field_behavior) = REQUIRED];
 // The unprotected JWS header values.
 google.protobuf.Struct header = 3;
}
```

This architecture ensures Kite agents participate as first class citizens in every major agent protocol while maintaining their unique capabilities.

3.3.4 Programmable Micropayment Channels

Direct money transfers work for occasional payments but fail catastrophically for agent patterns. Agents make thousands of tiny, frequent calls—messages, API requests, inference queries. Traditional payments would cost more in fees than value transferred. State channels transform this impossible economy into a profitable reality.

A state channel enables two parties to transact off chain with just two on chain transactions: open to lock funds and close to settle. Between these anchors, parties can exchange thousands of signed updates instantly. This yields high throughput, low latency, and fees amortized across up to millions of interactions. AI inference requiring sub hundred millisecond latency at $1 per million requests becomes economically viable only through programmable micropayment channels.

3.3.4.1 Channel Variants for Every Pattern Kite implements multiple channel types optimized for different interaction patterns:

Unidirectional channels flow value from user to merchant for simple metering. Perfect for API consumption, data feeds, and inference requests where value flows one direction.

Bidirectional channels enable refunds, credits, and two way value exchange. Services can pay agents for data, agents can receive rebates, errors trigger automatic refunds.

Programmable escrow channels embed custom logic in state transitions. EVM developers write arbitrary rules: conditional releases, multi party splits, time locked vesting. The channel becomes a mini smart contract.

Virtual channels route value through intermediaries without new on chain contracts. Agent A pays Agent C through hub B, enabling network effects without setup overhead.

Privacy preserving channels keep interactions confidential. Only channel opens and closes appear on chain. Thousands of micropayments remain private between participants, protecting competitive intelligence and usage patterns.

Net effect: every message becomes a payment with sub-cent precision, instant finality, and minimal on chain footprint. Perfect for agent pay per use and streaming economics.

3.3.4.2 How State Channel Limitations Become Advantages in Agent Economy State channels carry known limitations from their Ethereum origins. But agent use patterns transform these bugs into features.

Open/Close Channel Overhead
Traditional users rarely open and close channels, making setup costs prohibitive. But agents send hundreds of inferences to the same service over minutes. Setup costs amortize perfectly across concentrated bursts of activity. The "overhead" becomes negligible.

Liveness Assumption
Channels assume participants stay online to contest disputes. Human users go offline unpre-

dictably,but professional services with reputation at stake won't risk fraud for micropayments. It is therefore in the best interests of these services to maintain high availability.

Griefing Attacks
Malicious participants have the ability to force expensive dispute responses. But in Kite, agents and services maintain reputation scores. Griefing is therefore economically irrational, as it destroys reputation for minimal gain.

Predefined Participant Sets
Channel membership cannot change without closing the channel on chain. But agent interactions naturally have fixed participants, so the inability to change participants on an active channel only serves to secure agent payments further. User, agent, service—the relationships are determined and fixed at task initiation.

Parallel Transaction Processing
Channels process updates sequentially, working best for turn based applications. Agent interactions are inherently turn based: request, response, request. The limitation perfectly matches the use case.

3.3.5 Comparative Analysis: Cost and Latency Advantages

The architectural innovations within Kite, specifically Programmable Micropayment Channels and the Dedicated Stablecoin Payment Lane in the future, present a paradigm shift away from the economic and performance limitations of traditional blockchain payment systems. This section provides a comparative analysis focusing on two critical metrics: cost and latency.

3.3.5.1 Programmable Micropayment Channels Micropayment channels fundamentally transform the economics of blockchain transactions by decoupling on-chain settlement from off-chain state progression. This separation enables scalability and cost efficiency that traditional Layer-1 architectures cannot achieve.

Cost Analysis Qualitative Advantage:
Conventional Layer-1 blockchains impose a *per-transaction* cost model—every payment, regardless of value, must be individually broadcast and confirmed on-chain, incurring a full transaction fee (gas). For high-frequency or low-value transactions, this model quickly becomes prohibitive, as fees often exceed the value being transferred.

Kite's architecture replaces this with an **amortized cost model**. Channels incur on-chain fees only twice: once when the channel is opened and once when it is closed. All intermediate interactions occur off-chain as cryptographically signed state updates with near-zero computational overhead. This structure collapses the cost curve, allowing millions of micro-interactions to be settled economically under a single pair of on-chain transactions.

Quantitative Estimation Conventional On-Chain Transactions

- **Cost per payment:** Volatile and congestion-dependent, typically ranging between $0.10 and $50.00+.
- **Implication:** Sub-dollar payments become uneconomical; transaction fees often exceed principal value, rendering micropayments impractical on most Layer-1 networks.

Kite Micropayment Channel

- **On-Chain Cost:** Fixed, one-time cost to open and close a channel (approximately $0.01 on an optimized network).
- **Off-Chain Cost:** Effectively $0 per transfer, limited only by minimal local signing computation.
- **Amortized Cost per Payment:** For one million off-chain payments per channel, the effective cost per transaction becomes: $0.01 / 1,000,000 = $0.00000001
- This model enables economically viable, real-time micropayments and streaming interactions, previously unattainable under traditional blockchain economics.

Latency Analysis

Micropayment channels not only reduce cost but also redefine the temporal dynamics of transaction finality. Traditional blockchains are bound by block confirmation latency and probabilistic finality. A transaction is considered settled only after inclusion in a block and sufficient subsequent confirmations to prevent reorganization. This process, dictated by global consensus protocols, introduces unavoidable delays measured in seconds or even minutes.

In contrast, Kite's off-chain channels achieve deterministic and near-instantaneous settlement. State updates are peer-to-peer messages validated through cryptographic signatures rather than network-wide consensus. Each update's correctness is self-evident, bounded only by network round-trip time between participants. The result is a transactional experience that operates at machine speed rather than blockchain cadence.

Estimation System	Latency Characteristics	Typical Finality
Traditional Blockchains	Dependent on confirmation depth; inherently probabilistic.	2 seconds – several minutes
Kite Micropayment Channels	Peer-to-peer signature validation; deterministic finality between parties.	less than 100 milliseconds

Such sub-100 millisecond responsiveness enables interactive, streaming, and real-time economic models, allowing autonomous agents to transact continuously without perceptible delay.

More details on state channel and the latency analysis can be found in Link to paper.

3.4 Security with Programmable Trust

The promise of autonomous agents collapses without cryptographic security guarantees. Users cannot delegate real authority if agent compromise means unbounded loss. Services cannot accept agent requests without verifiable authorization. Regulators cannot approve agent operations without auditable compliance. Kite's security architecture provides mathematical certainty where traditional systems offer only promises.

3.4.1 Core Cryptographic Components

The Kite protocol achieves unbreakable security through three interlocking cryptographic primitives that create a complete chain of authorization from user intent to agent action.

3.4.1.1 Standing Intent: The Root of Authority The Standing Intent (SI) represents the user's cryptographically signed declaration of what an agent may do. This isn't a policy document or configuration file; it's a mathematical proof of authorization that cannot be forged or exceeded.

```
SI = sign_user(
    iss: user_address,      // Issuer: the authorizing user
    sub: agent_did,         // Subject: the authorized agent
    caps: {                 // Capabilities: hard limits
        max_tx: 100,        // Maximum per transaction
        max_daily: 1000     // Maximum daily aggregate
    },
    exp: timestamp          // Expiration: automatic revocation
)
```

The Standing Intent signed with the user's private key becomes the immutable root of trust. Every subsequent operation must trace back to a valid SI. The capabilities define mathematical boundaries that cannot be exceeded regardless of agent behavior, model hallucination, or service compromise. The expiration ensures forgotten authorizations cannot persist indefinitely.

3.4.1.2 Delegation Token: Agent Authorization The Delegation Token (DT) enables agents to authorize specific sessions for particular operations without exposing their permanent credentials:

```
DT = sign_agent(
    iss: agent_did,              // Issuer: the delegating agent
    sub: session_pubkey,         // Subject: the session key
    intent_hash: H(SI),          // Link: hash of Standing Intent
    operation: op_details,       // Scope: specific operation
    exp: now + 60s               // Expiration: short lived
)
```

The delegation token cryptographically proves the agent authorized this session for this operation within the Standing Intent's boundaries. The hash linkage ensures the agent cannot exceed user defined limits. The short expiration minimizes exposure from compromised sessions. The operation scope prevents session reuse for unauthorized actions.

3.4.1.3 Session Signature: Execution Proof The Session Signature (SS) provides the final cryptographic proof for transaction execution:

```
SS = sign_session(
    tx_details,  // Complete transaction data
    nonce,       // Replay prevention
    challenge    // Freshness proof
)
```

Services verify all three signatures before accepting operations. The Standing Intent proves user authorization. The Delegation Token proves agent delegation. The Session Signature proves current execution. This triple verification makes unauthorized actions cryptographically impossible rather than merely prohibited, assuming blockchain safety guarantees are met.

3.4.2 Provable Security Properties

The protocol provides mathematical guarantees about system behavior under adversarial conditions, transforming security from trust based to proof based.

Theorem 1: Bounded Loss
Theorem 1 (Multi-provider bound)**.** Let a Standing Intent SI be issued with capabilities $\mathcal{C}$ (including per-transaction and per-day limits) and duration D (days). Assume N authorized providers. At provider i, the daily cap is $\mathcal{C}_i$.max_daily and the SI expires after $D_i \leq D$ days. Then the maximum extractable value under complete agent compromise satisfies

$$\text{MEV} \;\leq\; \sum_{i=1}^{N} \mathcal{C}_i.\text{max\_daily}\; D_i.$$

If a global mechanism also caps aggregate daily spend by $\mathcal{C}$.max_daily_global, then

$$\text{MEV} \;\leq\; \min\Big(\,\mathcal{C}.\text{max\_daily\_global}\; D,\; \sum_{i=1}^{N} \mathcal{C}_i.\text{max\_daily}\; D_i\Big).$$

Proof sketch. Each transaction must carry a valid, non-expired SI. Provider-side enforcement guarantees that no transaction exceeds $\mathcal{C}_i$.max_tx and that the day's aggregate at provider i does not exceed $\mathcal{C}_i$.max_daily. Since SI at provider i expires after D_i days, total extraction at provider i is bounded by $\mathcal{C}_i$.max_daily D_i. Summing over $i = 1, \ldots, N$ yields the first inequality. If a global mechanism additionally enforces the per-day aggregate cap $\mathcal{C}$.max_daily_global, then over D days the total is at most $\mathcal{C}$.max_daily_global D. Taking the minimum of these independent bounds gives the second inequality. □
Corollary 1 (Single-provider)**.** For $N = 1$ with daily cap $\mathcal{C}$.max_daily and duration D,

$$\text{MEV} \;\leq\; \mathcal{C}.\text{max\_daily}\; D.$$

Interpretation. This yields precise risk quantification. For example, an authorization with a $100 daily cap for $D = 30$ days implies worst-case exposure of at most $3,000, even under full agent compromise.

Theorem 2: Unforgeability
Theorem 2. Without access to the user's private key, an adversary cannot create a valid SI for unauthorized agents.

Proof sketch. Standing Intent validity requires signature verification against the user's public key (e.g., ECDSA or EdDSA). Under the standard EUF-CMA assumption, an adversary without the private key cannot produce a valid signature on a new SI with non-negligible probability; observing prior intents, transcripts, or compromised agents does not enable minting new authorizations. □

Additional Security Properties The additional security properties include:

Forward Secrecy. Compromising a session key reveals only that session's operations; past and future sessions remain secure due to independent key generation.

Principle of Least Privilege. Each layer of delegation reduces authority: sessions have less power than agents, agents less than users. Authority only flows downward.

Automatic Expiration. All authorizations include expiration timestamps. Forgotten delegations cannot persist indefinitely; time becomes an automatic revocation mechanism.

Non-Repudiation. Cryptographic signatures provide undeniable proof of authorization. Users cannot deny authorizing agents, agents cannot deny authorizing sessions, and sessions cannot deny executing transactions.

3.4.3 Revocation Mechanism

Authorization without revocation is incomplete security. The protocol implements a comprehensive revocation system with multiple enforcement layers ensuring compromised or misbehaving agents cannot continue operating.

3.4.3.1 Immediate Local Revocation Users broadcast revocation messages to Kite Hub, which instantly propagates to all registered providers. This peer to peer propagation ensures near instantaneous revocation without waiting for blockchain confirmation. Services receiving revocation notices immediately reject all requests from revoked agents.

The revocation message contains:

- Agent identifier being revoked
- Revocation timestamp
- User signature proving authority
- Optional reason code for analytics

Network effects amplify revocation speed. Popular services with many connections propagate faster. Critical services prioritize revocation messages. The entire network typically learns of revocations within seconds.

3.4.3.2 Cryptographic Revocation Users sign revocation certificates that providers verify against:

```
RC = sign_user(
    action: "revoke",
    agent: agent_did,
    standing_intent: SI_hash,
    timestamp: now,
    permanent: true/false
)
```

Services check revocation certificates before accepting requests. Cached certificates persist across restarts. Permanent revocations cannot be reversed, providing strong security guarantees.

3.4.3.3 Economic Revocation Slashing conditions in agent bonds create economic incentives against continued operation post revocation:

Agent Bonds: Agents stake tokens as collateral for good behavior. Bonds are proportional to authorization limits, creating aligned incentives.

Slashing Triggers: Operating after revocation triggers automatic slashing. The protocol detects post revocation transactions and burns staked bonds.

Reputation Impact: Slashed agents suffer permanent reputation damage. Future authorizations require higher bonds. Services may refuse slashed agents entirely.

Distribution: Slashed funds are distributed to affected parties. Users recover losses from misbehaving agents. Services receive compensation for processing invalid requests.

This economic layer ensures that even if cryptographic and network revocations fail, agents have strong financial incentives to respect revocations. The cost of ignoring revocation exceeds any potential gain from continued operation.

3.4.3.4 Graceful Degradation in Adversarial Conditions The revocation system degrades gracefully under various failure modes:

Network Partition: Local revocation continues within network segments. Cryptographic certificates provide eventual consistency.

Hub Failure: Peer to peer propagation continues without central coordination. Services share revocation information directly.

Blockchain Congestion: Off chain revocation mechanisms operate independently. On chain slashing provides eventual enforcement.

Service Offline: Cached revocations persist. Services enforce revocations immediately upon restart.

This multi layer approach ensures revocation remains effective even under adversarial conditions or system failures. Users maintain ultimate control over their agents regardless of infrastructure state.

4. Agent Flows

The lifecycle of agent interactions encompasses three critical phases: authorization establishment, continuous communication, and value exchange through payments. Each phase builds on cryptographic foundations while maintaining intuitive developer experiences and user control.

4.1 Agent Authorization Flow

Authorization transforms human identity into agent capability through a carefully orchestrated flow that bridges traditional web authentication with blockchain settlement.

4.1.1 The Authorization Challenge

The flow begins when an agent attempts to access a service without valid credentials. The service returns a 401 Unauthorized response, indicating required authentication methods. This triggers the

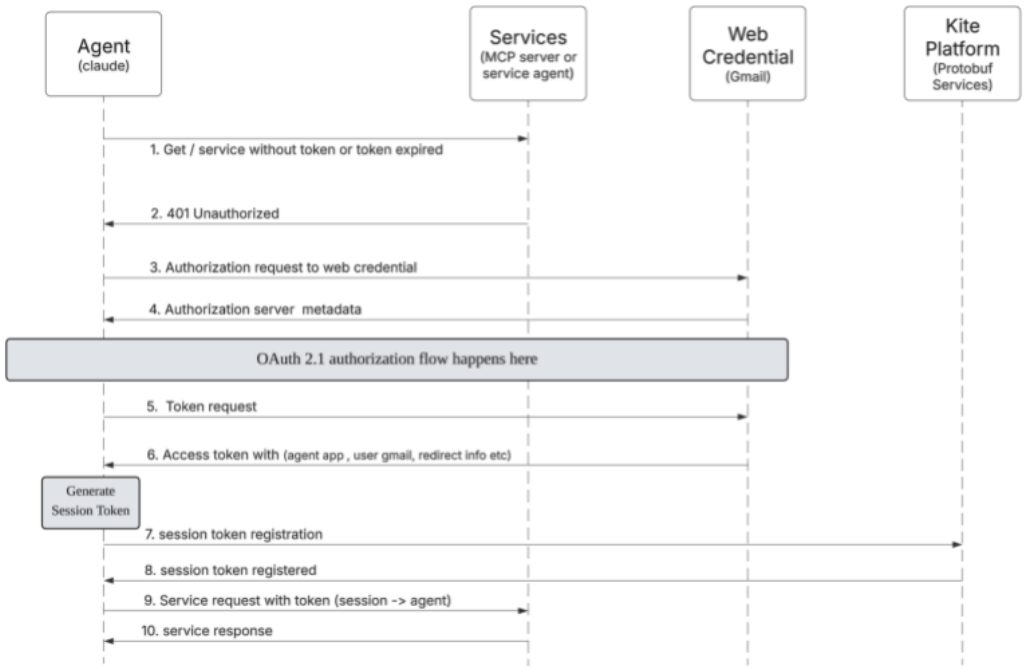

Fig. 5: Illustration of Agent Authorization Flow

authorization sequence that converts one time human authentication into persistent agent capability.

Key insight: Gmail/OAuth proves the user's web identity (represented as did:kite_chain_id:claude:scott for example, bundling Claude App with Scott's Gmail ID). The Kite session token transforms this one time proof into a time bounded, policy guarded capability the agent can use safely without repeatedly exposing the user's primary credentials.

4.1.2 Authorization Actors and Sequence

Four actors participate in the authorization flow:

- **Agent** (e.g., Claude): The AI system requesting service access
- **Service** (MCP server or service agent): The resource being accessed
- **Web Credential Provider** (Gmail): The identity verification source
- **Kite Platform and Chain**: The authorization and settlement layer

The authorization sequence proceeds through six critical steps:

Step 1: Initial Request Failure
The agent calls a service without a valid token or with an expired one. The service replies with 401 Unauthorized, initiating the authorization process.

Step 2: Discovery Phase
The service indicates it requires web credentials and points the agent to Gmail or other supported

providers. The agent retrieves the authorization server metadata through standard discovery mechanisms.

Step 3: OAuth Authentication
Standard OAuth 2.1 flow executes with Gmail. The user signs in and provides consent. The agent performs the token request and receives an access token bound to the agent application, user's Gmail identity, and redirect details. This creates cryptographic proof that a real human authorized this specific agent.

Step 4: Session Token Registration
After obtaining web credentials, the agent generates a local session key and registers a session token with Kite Platform and Chain. This registration binds:

- Agent and application identity
- Scopes and allowed operations
- Quotas and time to live (TTL)
- Proof chain back to user authorization

The session private key never leaves the local environment. A DID session registration call ensures the session is registered, linked, and recognizable by other services across the network.

Step 5: Service Retry
The agent retries the original service call, now presenting the session token signed by the ephemeral session key.

Step 6: Verification and Execution
The service verifies the token and checks it against Kite's registry and policies. If the token is valid, policies and quotas pass, and the token falls within its time window, the service executes the request and returns a successful response.

4.1.3 JWT Token Structure

When a session key is authorized and created, a corresponding JWT token encapsulates all necessary authorization information:

```
{
  "agent_did": "did:kite:alice.eth/chatgpt/assistant-v1",
  "apps": ["chatGPT", "cursor"],
  "timestamps": {
    "created_at": 1704067200,
    "expires_at": 1704070800
  },
  "proof_chain": "session->agent->user",
  "optional": {
    "user": "did:kite:alice.eth",
    "reputation_score": 850,
    "allowed_actions": ["read", "write", "pay"]
  }
}
```

Once created, subsequent calls to any service in the network contain both the JWT token and the session public key used for encryption. This dual credential system ensures both authorization and authenticity.

4.2 Agent Reputation and Trust Accumulation

Existing blockchains treat all accounts as equals. A newly created address possesses identical capabilities to one with years of legitimate history. This egalitarian approach fails catastrophically for agent systems where behavioral history should determine authorization scope.

4.2.1 Trust Dynamics for Agent Systems

Agent systems require sophisticated trust dynamics that traditional blockchains cannot provide:

Progressive Authorization: New agents must start with minimal permissions, perhaps $10 daily limits and restricted service access. Each successful operation contributes to reputation scores, automatically expanding capabilities.

Behavioral Adjustment: Successful operations should increase authorization over time. A reliable agent might see limits increase from $10 to $100 to $1,000 as trust accumulates. Conversely, violations should trigger automatic constraint tightening, reducing limits or requiring additional verification.

Trust Portability: Trust must transfer across services. An agent with established reputation on one platform should bootstrap trust on new platforms without starting from zero.

Verification Economics: Without native reputation, every agent interaction starts from zero trust, forcing expensive verification for routine operations. This overhead makes micropayments economically impossible.

4.2.2 Proof Chain Architecture

Kite provides a complete proof chain from session to agent to user to reputation, all verified by credited authorities. This chain enables instant trust verification without repeated authentication:

Service providers and users leverage this proof chain to make authorization decisions based on verifiable history rather than blind trust. For example, a user might grant:

- Read access to agents with reputation > 100
- Write access to agents with reputation > 500
- Payment authority to agents with reputation > 750
- Unlimited access to agents with reputation > 900

This graduated trust model enables safe progressive autonomy while maintaining security.

4.3 Agent Communication Flow

Human transactions occur in isolation. Agent operations require continuous communication and coordination. This fundamental difference demands native support for persistent connections, multi party coordination, and verifiable message exchange.

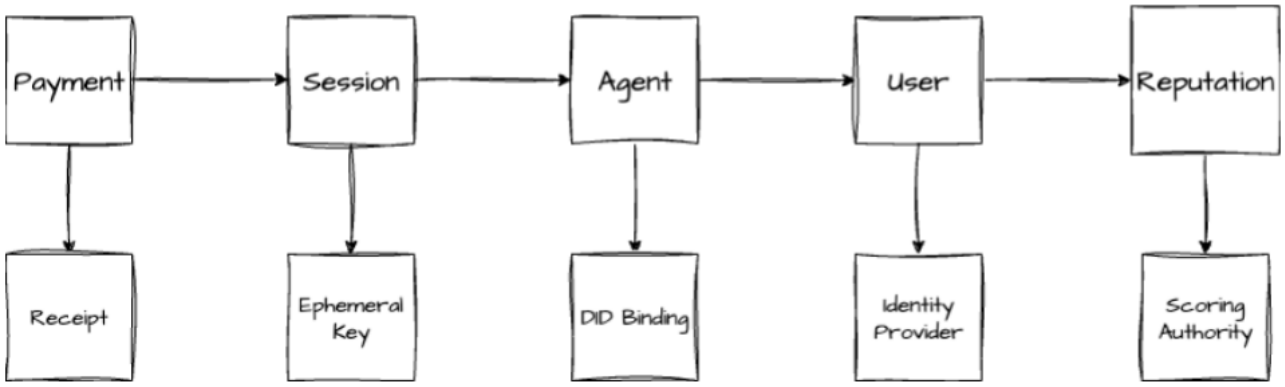

Fig. 6: Illustration of Kite's Proof Chain Architecture for Traceability and Accountability

4.3.1 Agent to Agent Messaging (A2A)

Encrypted channels enable negotiation, discovery, and coordination without exposing strategies or private information. The A2A protocol provides structured communication through Agent Cards that serve as the source of truth for capabilities and endpoints.

Agent Card Structure:

```
AgentCard {
  agent_did: "did:kite:alice.eth/gpt/trader",
  capabilities: ["streaming", "push_notifications"],
  security_schemes: ["JWT", "session_key"],
  endpoints: {
    primary: "wss://agent.example.com",
    fallback: "https://agent.example.com/api"
  },
  auth_methods: ["oauth", "did_auth"],
  session_scheme: "JWT + session_public_key"
}
```

Peers fetch Agent Cards to learn capabilities and support security schemes. The card declares endpoint URLs, authentication methods, and method availability. Most critically, it includes the Agent DID and session security scheme so any A2A compatible peer understands how to validate session scoped credentials.

4.3.2 Service Level Agreements

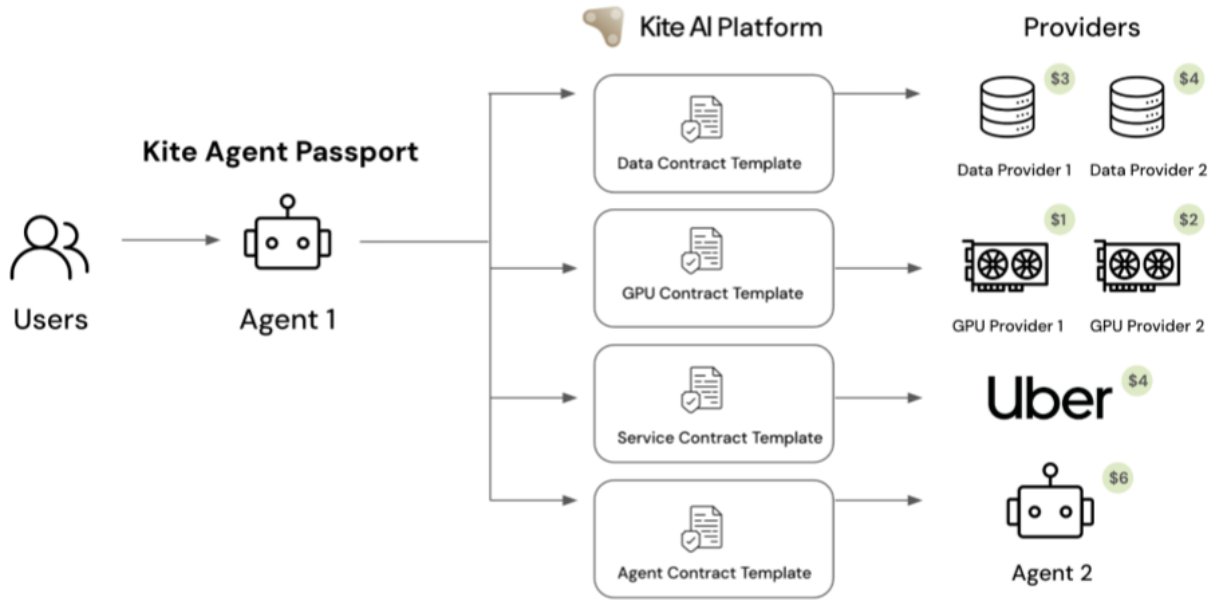

Fig. 7: SLA of Kite AI

Programmable contracts enforce response times, availability guarantees, and performance metrics with automatic penalties. Unlike traditional SLAs based on legal enforcement, these execute automatically through smart contracts:

- **Response Time**: Service must respond within 100ms or face automatic penalties
- **Availability**: 99.9% uptime with pro rata refunds for downtime
- **Accuracy**: Error rates below 0.1% or reputation slashing
- **Throughput**: Guaranteed 1,000 requests per second minimum

These programmable SLAs transform vague promises into code as law guarantees. Those SLA metrics:latency, uptime, accuracy, throughput,are measured off-chain by service telemetry and monitoring. To make them enforceable on-chain, an oracle submits signed attestations (or zk/TEE proofs) that translate the off-chain readings into on-chain facts. Contracts then evaluate those oracle reports to trigger refunds, penalties, or reputation slashing.

4.3.3 Reputation Networks

Persistent identity and performance tracking enables trust building across multiple interactions and services. Every interaction contributes to global reputation:

- Successful payments increase scores

- Fast responses boost reputation
- Failed deliveries decrease trust
- SLA violations trigger penalties

Reputation becomes portable across the entire network, solving the cold start problem where new relationships begin from zero trust.

4.4 Agent Payment Flow

Traditional payment rails were built for humans; Kite reimagines them for agents. In Kite's system, agents open lightweight state channels with services, where each interaction carries a cryptographically signed micro-voucher. A single channel close nets all balances on-chain, amortizing fees across millions of calls. This architecture enables genuine micropayments; around $0.000001 per message; alongside continuous, per-second or per-token streaming payments. Smart accounts and session keys enforce automatic quotas and instant revocation. The result is real-time, usage-based pricing that behaves like network packets rather than monthly billing cycles.

4.4.1 Channel Opening

The smart account deposits a limit (e.g., $50) into the channel contract, recording:

```
{
  merchant_did: "did:kite:service.ai/api",
  session_did: "did:kite:alice.eth/gpt/session-xyz",
  expiry: 1704070800,
  challenge_window: 3600  // 1 hour dispute period
}
```

This deposit acts as a bond for honest behavior. Both parties collectively sign a state update to initialize the channel's state. After initialization, they can transact quickly and freely off chain without blockchain overhead.

4.4.2 Pay Per Interaction (Off Chain)

Each message or API call carries a voucher signed by the session key:

```
{
  channel_id: "0xchan...",
  cumulative_amount: "10.523",  // Running total in USDC
  nonce: 4567,                   // Monotonic counter
  timestamp: 1704068000,
  signature: "0xsig..."
}
```

The channel tracks old state and new state, requiring all participants to agree on state changes. The merchant accepts work if the voucher validates and falls within quota and TTL. Streaming uses many small increments. Policies including spend caps, allowlists, and rolling windows are enforced locally by the Kite sidecar before voucher production.

4.4.3 Channel Settlement

Channels close through either cooperative or disputed paths:

Cooperative Close (Happy Path):
Parties co-sign the final state and submit once. Unused funds immediately return to the user. Gas costs are minimal. Settlement is instant.

Disputed Close:
Either party posts their latest signed state to the blockchain. The counterparty can challenge within the window by presenting a newer state, ensuring only the most recent update settles. If either party misbehaves, the other can trigger the on chain contract to close the channel and distribute funds.

Common dispute scenarios include:

- Participants going offline and failing to propose state transitions
- Participants refusing to co-sign valid state updates
- Participants attempting finalization with outdated states
- Participants proposing invalid state transitions

This dispute mechanism provides trustlessness, ensuring honest parties can exit deposits at any point regardless of counterparty behavior. The blockchain serves as the ultimate arbiter, guaranteeing that honest behavior is always rewarded and malicious behavior always punished.

4.4.4 Economic Transformation

This payment architecture enables an economic model impossible with traditional infrastructure:

Granular Pricing: Services price at the level of individual API calls, inference tokens, or compute milliseconds. A conversation might cost $0.02 in language model inference, $0.001 in embedding generation, and $0.0001 in storage.

Real Time Economics: Prices adjust dynamically based on load, priority, and resource availability. Peak hours might cost more. Premium models command higher rates. Urgent requests pay priority fees.

Automatic Budgets: Agents operate within cryptographically enforced spending limits. When budgets approach limits, agents automatically throttle or halt operations.

Instant Settlement: No waiting for monthly bills or reconciliation. Every interaction settles immediately. Services receive payment instantly. Users see real time spending.

Shifting from billing cycles to packet-level economics unlocks models legacy rails can't support: every message settles as a payment, every payment is programmable, and every program is verifiable on-chain. This is the economic base layer for autonomous agents.

5. Comparison With Existing Approaches

While various projects attempt to address pieces of the agent payment puzzle, Kite provides the first complete infrastructure designed from first principles for autonomous agent operations. Understanding how we surpass existing approaches illuminates why partial solutions cannot enable the agent economy.

5.1 Beyond Tempo: From Payments-First to Agent-Native

Tempo represents the current state-of-the-art in blockchain payment optimization, yet it remains fundamentally limited by human-centric design assumptions. While Tempo achieves improvements in payment efficiency, Kite delivers the agent-native infrastructure required for autonomous operations.

Where Tempo Stops, Kite Begins:

Tempo optimizes for human-initiated payments with stablecoin fees and dedicated lanes. This solves important problems but misses the fundamental requirements of agent systems. Kite goes dramatically further:

Hierarchical Identity Architecture: While Tempo treats all actors equally, Kite implements three-tier identity (user $\rightarrow$ agent $\rightarrow$ session) with cryptographic delegation. This isn't a feature addition; it's a fundamental reimagining of how authority flows in autonomous systems.

Programmable Constraint Enforcement: Tempo assumes humans evaluate each transaction. Kite recognizes that agents will hallucinate, err, and occasionally malfunction. Our smart contract constraints provide mathematical guarantees that agents cannot exceed defined boundaries, regardless of model state or compromise.

Native Agent Transaction Types: Tempo processes payments. Kite embeds computation requests, API calls, and data queries directly in transactions. Every message becomes a billable event with cryptographic proof of authorization and delivery.

Protocol-Level Integration: While Tempo remains isolated, Kite provides native compatibility with AP2, A2A, MCP, OAuth 2.1, and emerging agent standards. Agents operate across ecosystems without adaptation.

Tempo improved blockchain payments. Kite created agent-native infrastructure. The difference is fundamental, not incremental.

5.2 Native Interoperability vs Fragmented Standards

Agent Commerce Kit (ACK), Google's AP2, and ERC-8004 define useful standards, Kite provides the execution layer to make these visions operational. Kite doesn't just comply with these standards; we provide the infrastructure that makes them work.

5.2.1 Making ACK Operational

ACK defines principles for agent commerce. Kite implements them:

- **Agent Passport**: Not just identity but complete credential management with selective disclosure

- **Programmable Payment Lanes**: Not just payments but agent-native transaction types
- **Intent-Based Authorization**: Not just policies but cryptographically enforced constraints

We transform ACK's specifications into working infrastructure.

5.2.2 Extending Google's AP2

AP2 introduces intent mandates for agent payments. Kite surpasses this with:

- Three-tier identity hierarchy providing granular delegation
- State channels enabling micropayment streaming AP2 cannot achieve
- SLA contracts creating enforceable service guarantees beyond simple mandates

5.2.3 Implementing ERC-8004's Vision

ERC-8004 proposes registries for identity, reputation, and validation. Kite delivers:

- Agent passports with verifiable credentials exceeding identity requirements
- Behavioral reputation accumulated across all interactions
- Cryptographic proof chains providing indisputable validation

5.3 The Complete Solution

Other projects solve fragments of the problem. Tempo improves payments but ignores agents. SkyFire enables agents but creates systemic risk. Standards define interfaces but lack implementation.

Kite provides the complete stack: from blockchain optimized for agent patterns to identity systems designed for delegation, from programmable constraints that prevent hallucination losses to state channels enabling micropayment streaming, from native protocol compatibility to cryptographic security guarantees.

We didn't build a better payment system. We built the infrastructure for the agent economy.

6. Use Cases

The convergence of agent autonomy and programmable payments enables business models impossible with traditional infrastructure. These use cases demonstrate the transformative potential when every interaction becomes a transaction.

6.1 Gaming: True Microtransactions at Global Scale

Current reality: Buying a $0.50 skin or $1 extra life involves credit card rails charging $0.30 plus 2.9% fees, multi-day settlement, and foreign exchange markups. Small transactions become economically impossible.

With Kite's Agent-Native Rails:

Players confirm purchases with one-click instant PYUSD payments. No card tokenization, no banking intermediaries, just cryptographic value transfer. A gamer in Brazil buys from a U.S. studio without foreign exchange markup or payment failures.

Game agents bundle multiple in-app purchases intelligently, enforce parental spending limits automatically, and apply loyalty credits programmatically. Parents set "maximum $20 monthly on games" and the blockchain enforces this limit absolutely. Game developers receive instant settlement, enabling real-time revenue sharing with creators.

This unlocks sub-dollar items that were previously economically unviable. Games can price items at their true value rather than bundling to meet payment minimums. The entire gaming economy becomes more liquid, transparent, and accessible.

6.2 IoT: Machine-to-Machine Bandwidth Markets

Current reality: IoT devices rely on monthly telco billing with flat plans or expensive roaming charges. A drone needing 5 minutes of bandwidth pays for a month of service.

With Kite's Agentic Payments:

Devices open per-second PYUSD payment channels to ISPs or satellite providers. Each packet consumed triggers a streaming micropayment. Sessions close automatically with instant reconciliation. A car downloads a map update for $0.02. A sensor uploads telemetry for $0.001. A drone streams video for $0.50 per minute.

Global clearing through stablecoin rails removes carrier foreign exchange and roaming complexity. Devices connect to any available network worldwide, paying exactly for consumption. This enables granular, pay-as-you-go bandwidth markets where machines pay only for what they use, making IoT deployment dramatically cheaper and more scalable.

6.3 Creator Economy: Direct Fan-to-Creator Revenue

Current reality: Creator revenue flows through ad networks and platforms taking 30% cuts. Minimum payouts of $5 to $10 create barriers. Settlement takes weeks.

With Kite's Stablecoin-Native Payments:

Fans send sub-cent PYUSD tips in real time. A viewer might pay $0.01 every 10 seconds while watching a stream, creating continuous revenue flow. Agents aggregate micro-tips into transparent, programmable splits: 70% creator, 20% platform, 10% moderator. No minimums, no waiting periods, just instant value transfer.

Creators receive payments globally without banking relationships. A musician in Nigeria reaches fans in Japan without international wire fees. This redefines the fan-to-creator relationship, creating direct, continuous revenue streams instead of aggregated monthly payouts.

6.4 API Economy: Every Call Becomes a Transaction

Current reality: API billing uses monthly invoices or prepaid credits. Opaque pricing, manual reconciliation, and credit risk create friction.

With Kite's Agent-to-Agent Billing:

Each API call meters in real time with fractions of cents streamed per inference or request. An LLM endpoint charges $0.00001 per token. A weather API charges $0.0001 per query. A GPU cluster charges $0.01 per second of compute.

Agents open state channels, sign micro-vouchers per call, and settle once per session. Throughput remains high while fees approach zero. Usage becomes cryptographically auditable: providers prove what was consumed, clients enforce quotas automatically.

This model turns every API call into an autonomous, self-clearing financial transaction. No invoices, no collections, no credit risk. Just instant value exchange at machine speed.

6.5 E-Commerce: Programmable Checkout and Escrow

Current reality: E-commerce involves credit card authorizations, delayed settlement, and expensive chargebacks. Buyers have little recourse. Sellers face fraud risk.

With Kite's Agentic Escrow:

Shopping agents lock PYUSD into on-chain escrow instead of direct merchant payment. Funds release only when verifiers (delivery oracles, SLA agents) confirm conditions: "delivered," "service completed," or timeout without dispute.

Partial captures let merchants settle only shipped items. Refunds and voids execute programmatically and instantly. Platform fees split automatically on capture. This creates chargeback-free, programmable checkout where agents enforce business rules and users get verifiable recourse.

6.6 Personal Finance: Autonomous Money Management

Current reality: Financial management requires sharing bank passwords with apps or manual transaction entry. Automation is limited and risky.

With Kite's Controlled Delegation:

Users deploy AI agents to manage budgets, pay bills, and invest small amounts. Instead of bank passwords, agents hold controlled on-chain credentials with spending limits. The agent pays utility bills directly to service agents, streams micro-deposits into savings, and dollar-cost-averages into ETFs with $1 daily stablecoin transfers.

Every action ("Paid $50 to ElectricityCo," "Bought $20 of ETF XYZ") logs on-chain with cryptographic proof. Users audit instantly. Tax preparation becomes automatic. Agents become trustworthy money managers with programmable guardrails and micropayment capabilities banks cannot match.

6.7 Knowledge Markets: Decentralized Expertise

Current reality: Knowledge queries go to single APIs with opaque pricing and unknown quality. Expertise remains siloed and undermonetized.

With Kite's Collaborative Networks:

Users post complex queries spanning legal, medical, and research domains. Specialist agents collaborate on answers, with each contribution scored for value. Instant micropayments stream to contributors based on their impact. A legal agent might earn $0.50 for contract interpretation. A medical agent might earn $0.30 for symptom analysis. A research agent might earn $0.20 for citation verification.

All identities, contributions, and payouts verify on-chain, building transparent reputations. Instead of opaque APIs, users get crowdsourced expertise. Contributors earn fair, granular rewards without gatekeepers.

6.8 Supply Chain: Autonomous Commercial Networks

Current reality: Supply chains rely on EDI, manual approvals, and paper documentation. Settlement takes 30 to 90 days. Trust requires expensive verification.

With Kite's Enterprise Agents:

Manufacturers, suppliers, and shippers deploy agents with defined spending limits. When inventory runs low, agents negotiate orders, escrow payments, and coordinate logistics automatically. IoT oracles confirm milestones ("arrived at port," "passed inspection"), triggering conditional fund releases.

The result: 24/7 autonomous supply chains with instant settlement, programmable escrow, and tamper-proof audit trails. No EDI protocols, no manual approvals, no trust issues. Global commerce operates at the speed of software.

6.9 Decentralized Governance: AI-Enhanced DAOs

Current reality: DAOs struggle with voter apathy, slow decision-making, and inflexible multisig operations.

With Kite's AI Governors:

Organizations entrust AI agents with treasury operations including rebalancing, voting, and routine transactions, all under on-chain policy limits. Agents react instantly to market changes while major decisions still require human token-holder approval. An agent might rebalance treasury holdings within 5% bands autonomously but require DAO votes for larger shifts.

Every trade, vote, and action logs transparently on-chain. DAOs evolve into truly autonomous organizations, blending human governance with AI execution. Faster than multisigs, safer than black-box bots, and fully accountable through cryptographic proofs.

7. Future Work

As Kite evolves from infrastructure to ecosystem, several research and development initiatives will extend programmable trust and agent-native capabilities beyond payments into broader domains of verifiable computation and autonomous coordination.

7.1 Zero-Knowledge Verified Agent Credentials

Customized storage modules for zero-knowledge verified credentials could be developed that enable agents to prove attributes without revealing underlying data. This extends beyond simple identity to complex attestations about capabilities, certifications, and permissions.

ZK-Verified Attributes: Agents will prove age, KYC status, professional licenses, or security clearances without exposing personal information. A medical agent could prove board certification without revealing the doctor's identity.

Custom Storage Architecture: Encrypted credential storage with ephemeral decryption keys ensures that sensitive data remains protected even if storage is compromised. Credentials expire automatically, rotate regularly, and revoke instantly.

Composable Proofs: Multiple credentials combine into complex proofs. An agent might prove it represents an accredited investor over 18 with no sanctions exposure, all in a single zero-knowledge proof.

7.2 Verifiable Inference and Computation

We plan to introduce primitives for verifiable inference, proving not only what decision an agent produced but also linking it cryptographically to model parameters, prompts, and data sources.

Inference Verification: Smart contracts will verify that specific model outputs derived from specific inputs using specific model weights. This enables cryptographic proof of AI decisions for regulatory compliance.

Task Attestation: Complex multi-step agent tasks will generate proof chains showing each decision point. Disputes resolve through cryptographic evidence rather than testimony.

Data Availability Layer: Short-term storage of inference proofs using data availability protocols ensures low-cost retention without permanent blockchain bloat. Proofs remain queryable for compliance periods then expire automatically.

7.3 Portable Reputation Networks

Agents and services will accumulate portable, on-chain reputation based on transaction history, SLA performance, and peer attestations. This reputation becomes a first-class primitive in the ecosystem.

Behavioral Scoring: Every interaction contributes to reputation scores. Successful payments increase trust. Fast responses boost ratings. Failed deliveries decrease scores. The blockchain provides indisputable historical records.

Cross-Platform Portability: Reputation earned on one platform transfers to others through cryptographic proofs. An agent with high reputation on Kite gets preferential treatment everywhere.

Automated Trust Decisions: High-reputation agents gain automatic approval for higher limits, better rates, and premium access. Bad actors face automatic restrictions. Trust becomes programmable and responsive.

7.4 Verifiable Service Discovery

A trustless discovery layer could be implemented where agents locate compatible services through verifiable attestations rather than centralized directories.

Capability Attestations: Services prove their capabilities through cryptographic credentials. An LLM service might prove model size, training data, and performance benchmarks.

Compliance Proofs: Services demonstrate regulatory compliance through zero-knowledge proofs. A financial service could prove SEC registration without revealing corporate details.

Performance History: Past SLA performance becomes queryable on-chain. Agents discover not just what services claim to offer but what they actually deliver.

7.5 Comprehensive Traceability and Attestation

Kite will expand its traceability layer so that every agent action pairs with attestations from relevant parties, creating complete audit trails for complex multi-party interactions.

Cryptographic Lineage: Every payment, message, or inference links to its complete history. Auditors trace funds from origin to destination with mathematical certainty.

Regulatory Compliance: Verifiable records satisfy regulatory requirements without exposing competitive intelligence. Financial authorities verify compliance through cryptographic proofs rather than document requests.

Automated Recourse: Disputes link automatically to verifiable event chains. Smart contracts adjudicate based on cryptographic evidence. Resolution becomes programmatic rather than legal.

These initiatives transform Kite from payment infrastructure into a complete platform for verifiable autonomous computation. Agents will not only transact but also prove their decisions, build portable trust, and operate with complete transparency.

8. Conclusion

The agent economy is not a distant possibility but an immediate reality constrained only by infrastructure. Language models have achieved the sophistication to manage complex workflows, make nuanced decisions, and coordinate across systems. Enterprises have moved beyond pilots to production deployments. Regulators have established frameworks for algorithmic accountability. The only missing piece is infrastructure that treats agents as first-class economic actors.

8.1 The Infrastructure Imperative

Traditional systems fail agents at every level. Authentication assumes human users, forcing agents through OAuth flows designed for occasional logins. Payment rails charge $0.30 minimums on $0.01 transactions, making micropayments economically impossible. Trust models offer only binary choices between unlimited access and complete denial. These aren't bugs to patch but fundamental architectural mismatches.

Incremental improvements cannot bridge this gap. Adding API keys to existing systems doesn't solve the credential explosion problem. Reducing payment fees doesn't enable streaming micropayments.

Improving audit logs doesn't create cryptographic proof of compliance. The agent economy requires infrastructure reimagined from first principles.

8.2 Kite's Comprehensive Solution

Kite provides the complete infrastructure stack for autonomous agent operations:

Three-Tier Identity: User → Agent → Session hierarchy with cryptographic delegation ensures that authority flows safely from humans to agents to individual operations. Each layer provides bounded autonomy with mathematical guarantees.

Programmable Constraints: Smart contracts enforce spending limits, time windows, and operational boundaries that agents cannot exceed regardless of hallucination, error, or compromise. Code becomes law.

Native Payment Primitives: State channels enable micropayments at $0.000001 per message. Dedicated payment lanes guarantee predictable costs. Every interaction becomes a billable event with instant settlement.

Universal Interoperability: Native compatibility with A2A, MCP, OAuth 2.1, and emerging standards ensures agents operate across ecosystems without adaptation. One infrastructure, universal access.

Cryptographic Security: Triple-signature verification, bounded loss theorems, and multi-layer revocation provide mathematical certainty about system behavior. Trust becomes verifiable, not promised.

8.3 The Economic Transformation

The implications extend far beyond technical infrastructure. When every API call becomes a transaction, business models transform. When micropayments cost nothing, new economies emerge. When agents manage money autonomously, human productivity multiplies.

Gaming economies unlock items worth pennies. IoT devices pay for bandwidth by the packet. Creators receive continuous revenue streams. Knowledge markets reward expertise granularly. Supply chains operate continuously. DAOs govern intelligently. These aren't optimizations of existing models but entirely new forms of economic interaction.

The transformation has already begun. Early adopters report 100x payment cost reduction, 50x latency improvement, and business models previously impossible. But we're witnessing the earliest stages of a fundamental platform shift comparable to the emergence of the internet itself.

8.4 The Path Forward

Kite's roadmap extends beyond payments to complete verifiable computation. Zero-knowledge credentials will enable privacy-preserving attestations. Verifiable inference will prove AI decisions. Portable reputation will create universal trust. Service discovery will connect agents autonomously. Every interaction will become verifiable, traceable, and programmable.

Yet the foundation already exists. The blockchain is operational. The protocols are defined. The standards are implemented. Developers can begin building today on infrastructure that will scale to billions of agents executing trillions of transactions.

8.5 A Call to Action

The agent economy is not waiting for better models or faster compute. It's waiting for infrastructure that enables autonomous operation with safety guarantees. That infrastructure now exists.

For developers, Kite provides the primitives to build agent-native applications impossible on traditional platforms. For users, Kite transforms AI from chatbots into trusted financial actors.

The transition from human-mediated to agent-native infrastructure represents the next great platform shift. Just as mobile computing required rethinking every assumption about user interaction, the agent economy demands rethinking every assumption about identity, payment, and trust. The infrastructure for this transformation is no longer theoretical. It's operational, scalable, and ready for the applications that will define the next decade of computing. The agent economy begins with Kite.

The future isn't just automated. It's autonomous. And with Kite, it starts now.

Contributors

Authors

- Scott Shi, CTO, Kite AI.
- Zerui Cheng (Princeton University), research collaborator of Kite AI.
- Chen Xi (Uber Technology), research collaborator of Kite AI.
- Yi Huang, advisor of Kite AI.
- Lyon Li, member of technical staff, Kite AI.
- Uddhav Marwaha, Head of Payment, Coinbase.
- David Weber, Head of PyUSD, PayPal.
- Dr. Chi Zhang, CEO, Kite AI.

References

[1] Anthropic. (2024). Model Context Protocol (MCP).
Retrieved from https://www.anthropic.com/news/model-context-protocol

[2] BIP-32. (2012). Hierarchical Deterministic Wallets. Bitcoin Improvement Proposal. Retrieved from https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

[3] EIP-712. (2017). Ethereum typed structured data hashing and signing. Ethereum Improvement Proposal. Retrieved from https://eips.ethereum.org/EIPS/eip-712

[4] EIP-1559. (2021). Fee market change for ETH 1.0 chain. Ethereum Improvement Proposal. Retrieved from https://eips.ethereum.org/EIPS/eip-1559

[5] EIP-4337. (2023). Account Abstraction Using Alt Mempool. Ethereum Improvement Proposal. Retrieved from https://eips.ethereum.org/EIPS/eip-4337

[6] ERC-3009. (2020). Transfer With Authorization. Ethereum Request for Comments. Retrieved from https://eips.ethereum.org/EIPS/eip-3009

[7] ERC-4337. (2023). Account Abstraction via Entry Point Contract specification. Ethereum Request for Comments. Retrieved from https://www.erc4337.io/

[8] European Union. (2024). EU AI Act. Regulation on Artificial Intelligence. Retrieved from https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

[9] Google. (2025). Agent-to-Agent (A2A) Protocol Specification. Google AI Developer Documentation.

[10] Google. (2024). Agent Payments Protocol (AP2) Specification. Google AI Developer Documentation.

[11] IETF. (2015). RFC 7515: JSON Web Signature (JWS). Internet Engineering Task Force. Retrieved from https://datatracker.ietf.org/doc/html/rfc7515

[12] IETF. (2015). RFC 7519: JSON Web Token (JWT). Internet Engineering Task Force. Retrieved from https://datatracker.ietf.org/doc/html/rfc7519

[13] McKinsey Global Institute. (2023). The economic potential of generative AI: The next productivity frontier. McKinsey & Company.

[14] OAuth 2.1. (2023). The OAuth 2.1 Authorization Framework. Draft IETF Specification. Retrieved from https://datatracker.ietf.org/doc/draft-ietf-oauth-v2-1/

[15] Paradigm. (2025). Tempo: A Payments-First Blockchain Architecture. Paradigm Research.

[16] Protocol Buffers. (2024). Google's language-neutral, platform-neutral extensible mechanism for serializing structured data. Retrieved from https://protobuf.dev/

[17] W3C. (2022). Decentralized Identifiers (DIDs) v1.0. World Wide Web Consortium. Retrieved from https://www.w3.org/TR/did-core/

[18] W3C. (2022). Verifiable Credentials Data Model v1.1. World Wide Web Consortium. Retrieved from https://www.w3.org/TR/vc-data-model/

[19] Buterin, V. (2014). Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum White Paper.

[20] ECDSA. (2013). Elliptic Curve Digital Signature Algorithm. ANSI X9.62-2005. American National Standards Institute.

[21] EdDSA. (2017). Edwards-curve Digital Signature Algorithm. RFC 8032. Internet Engineering Task Force.

[22] ENS. (2017). Ethereum Name Service Documentation. Retrieved from https://docs.ens.domains/

[23] FATF. (2021). Updated Guidance for a Risk-Based Approach to Virtual Assets and Virtual Asset Service Providers. Financial Action Task Force.

[24] ISO 20022. (2022). Universal financial industry message scheme. International Organization for Standardization.

[25] Lightning Network. (2016). The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. Poon, J., & Dryja, T.

[26] MCC Codes. (2024). Merchant Category Codes. ISO 18245. International Organization for Standardization.

[27] Merkle, R. (1987). A Digital Signature Based on a Conventional Encryption Function. Advances in Cryptology — CRYPTO '87.

[28] Raiden Network. (2017). Fast, cheap, scalable token transfers for Ethereum. Retrieved from https://raiden.network/

[29] State Channels. (2018). Generalized State Channels on Ethereum. Miller, A., Bentov, I., Kumaresan, R., & McCorry, P.

[30] x402. (2024). Agent Payment Standard Specification. Industry Consortium Draft.

[31] Zero Knowledge Proofs. (1985). The Knowledge Complexity of Interactive Proof Systems. Goldwasser, S., Micali, S., & Rackoff, C.

[32] BLS Signatures. (2001). Short Signatures from the Weil Pairing. Boneh, D., Lynn, B., & Shacham, H.

[33] Travel Rule. (2019). IVMS101 - interVASP Messaging Standard. Joint Working Group on interVASP Messaging Standards.

[34] EUF-CMA Security. (1988). A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. Goldwasser, S., Micali, S., & Rivest, R.

[35] How do Apply Pay and Google Pay work? (2025).
Retrieved from https://bytebytego.com/guides/how-applegoogle-pay-works

[36] ERC8004: Trustless Agents. (2025). Retrieved from https://eips.ethereum.org/EIPS/eip-8004

[37] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. "Sprites and state channels: Payment networks that go faster than lightning." In International conference on financial cryptography and data security, pp. 508-526. Cham: Springer International Publishing, 2019.

Appendix A. Kite AI Tokenomics

A.1 The Mission of Kite AI

Kite AI is building **the first blockchain for agentic payments**, a foundational platform where autonomous AI agents can operate with verifiable identity, programmable governance, and seamless payments. The company's purpose-built Layer-1 blockchain and Agent Passport system enable AI agents to function as first-class economic actors, creating emergent capabilities through composable interactions. Founded by AI and data infrastructure veterans from Databricks, Uber, and UC Berkeley, Kite AI has raised **$33 million** from top-tier investors, including PayPal, General Catalyst, Coinbase Venture and leading blockchain foundations.

A.2 Kite AI Network Overview

The Kite AI blockchain is a **Proof-of-Stake (PoS) EVM-compatible Layer-1** chain that serves as a low-cost, real-time payment mechanism and coordination layer necessary for autonomous agents to interoperate. Alongside the L1 are **a suite of modules, modular ecosystems that expose curated AI services (e.g., data, models, and agents) to users**. Modules operate as semi-independent communities that interact with the L1 for settlement and attribution, while providing specialized environments tailored to particular verticals. Together, the L1 and Modules form a tightly coupled ecosystem where users may assume distinct roles, module owners, validators, or delegators.

KITE is the native token of the Kite AI Network, driving incentives, staking, and governance on Kite AI blockchain. The economics are designed to tie token value directly to real AI service usage while discouraging short-term extraction.

A.3 KITE Token Utilities

The utility of the KITE token will be rolled out in two phases: Phase 1 utilities are introduced at the time of token generation so that early adopters can immediately participate in the Kite network, while Phase 2 utilities will be added with the launch of the Mainnet.

Phase 1 Utilities (at the time of token generation).

- **Module Liquidity Requirements.** Module owners who have their own tokens must lock KITE into permanent liquidity pools paired with their module tokens to activate their modules. This requirement scales with module size and usage, creating deep liquidity while removing tokens from circulation. Liquidity positions are non-withdrawable while modules remain active, ensuring long-term token commitment from the most value-generating participants.
- **Ecosystem Access & Eligibility.** Builders and AI service providers must hold KITE to be eligible to integrate into the Kite ecosystem. This gives the KITE token immediate utility as an access token to participate in the agentic payment ecosystem and aligns the interests of businesses and builders with that of the Kite network as a whole.
- **Ecosystem Incentives.** A portion of the KITE supply will be distributed to users and businesses who bring value to the Kite ecosystem.

Phase 2 Utilities (at the launch of Mainnet).

- **AI Service Commissions.** The protocol collects a small commission from each AI service transaction and can swap it for KITE on the open market before distributing it to the module and the Kite L1. This ensures that service operators receive payment in their currency of choice while the module and L1 can receive native tokens that increase their stake and influence within the ecosystem. Protocol margins are converted from stablecoin revenues into KITE, creating continuous buy pressure tied directly to real AI service usage and revenues, ensuring token value scales with network adoption.
- **Staking.** Staking KITE secures the network and grants users eligibility to perform services in exchange for rewards. Module owners develop, operate, and manage a module on the Kite network, validators participate in a Proof-of-Stake consensus, and delegators stake tokens to secure and support modules they find value in.
- **Governance.**Token holders vote on protocol upgrades, incentive structures, and module performance requirements, aligning the interests of stakeholders with long-term network health.

A.3 Modules, Validators, and Delegators

Validators. Kite validators secure the network by staking tokens and participating in consensus. Each validator selects a specific module to stake on, so that incentives are aligned with that module's performance. Core responsibilities include maintaining network security and consensus stability, participating in governance and voting, and advancing community collaboration and ecosystem growth.

Delegators. Kite delegators stake tokens to secure the network. Each delegator must select a specific module to stake on, aligning their incentives with that module's performance. Delegators are expected to take part in ecosystem governance and engage with the community.

Continuous Rewards with "Piggy Bank". Modules, validators, and delegators receive token emissions through a novel continuous reward system designed to incentivize long-term holding. Participants accumulate KITE rewards over time in a "piggy bank" — they can claim and sell their accumulated tokens at any point, but doing so permanently voids all future emissions to that address. The mechanism transforms token recipients into long-term aligned stakeholders who must weigh immediate liquidity against ongoing value accrual.

A.4 Initial Allocation and Supply

The total supply of KITE is capped at 10 billion. The initial allocation is shown in Table 2.

Token Distribution.

- *Ecosystem and Community (48%).* Ecosystem and community tokens are dedicated to accelerating user adoption, builder and developer engagement, and ecosystem liquidity. They will fund community airdrops, liquidity programs, and growth initiatives that expand participation, reward meaningful contributions, and drive Kite's transition from launch to broad utility.

Table 2: Initial KITE token allocation.

Category	Share
Ecosystem and Community	48%
Investors	12%
Modules	20%
Team, Advisors, Early Contributors	20%
Total	**100%**

- *Investors (12%).* Investor tokens are distributed under structured vesting schedules to align investor interests with the network's long-term growth, ensuring that early financial supporters remain committed as the ecosystem expands.
- *Modules (20%).* Tokens allocated to modules will be used to incentivize the development of high-quality AI services and to expand the infrastructure that enables users to interact seamlessly with the Kite ecosystem. These funds will support developer grants, performance-based rewards, and the build-out of services that enhance both the intelligence and accessibility of the network.
- *Team, Advisors, Early Contributors (20%).* Team and advisor tokens align the long-term incentives of Kite's builders with the network's sustained success. These allocations reward early contributors, developers, and strategic advisors who are building and scaling the network, with multi-year vesting schedules that promote stability and accountability as the ecosystem matures.

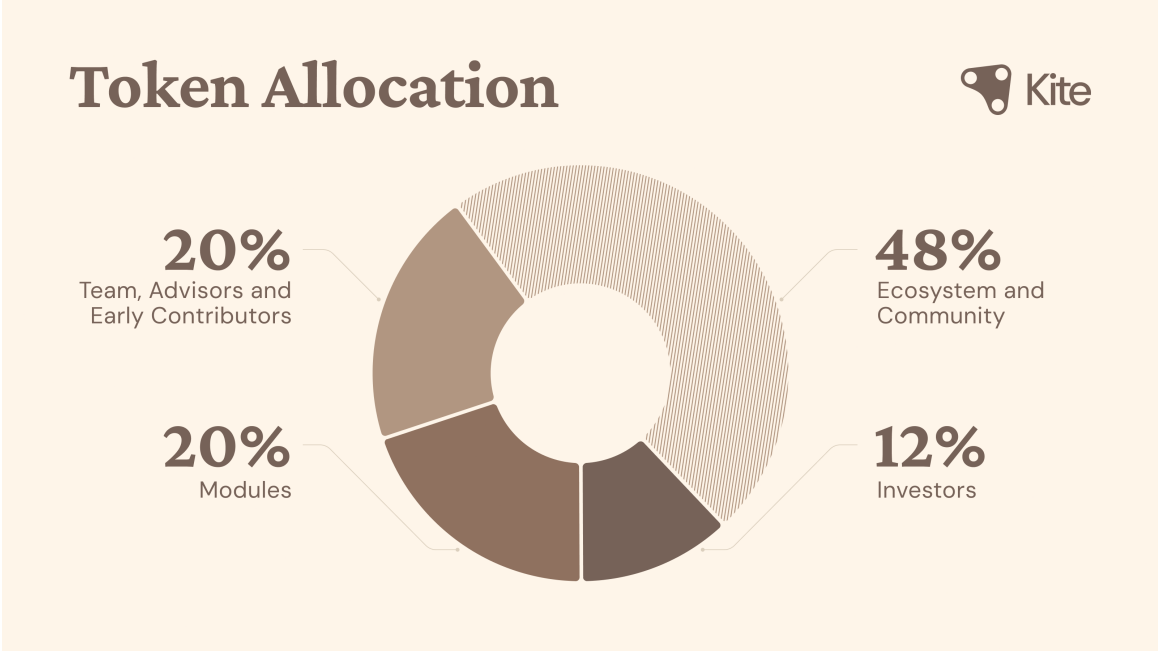

Fig. 8: Illustrative KITE allocation breakdown.

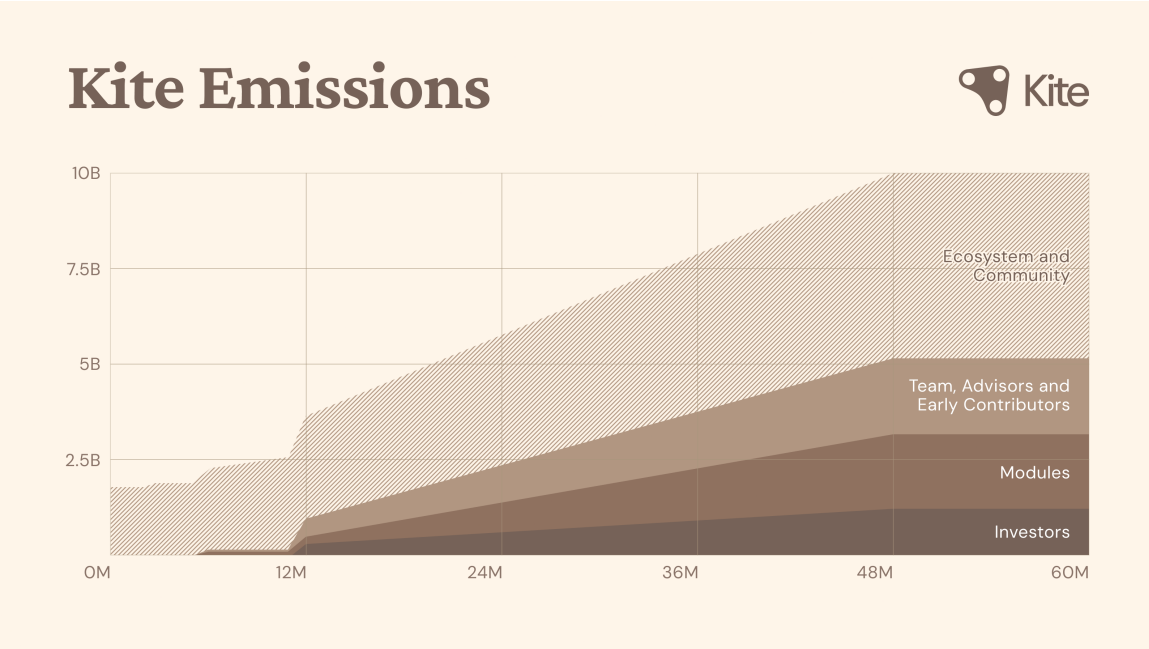

Fig. 9: Illustrative KITE token emission along time.

A.5 The Kite Network Value Capture

Revenue-Driven Network Growth. Kite AI implements multiple mechanisms that directly tie token value to network revenues and usage. A percentage of fees from AI service transactions are collected as commission for modules and the network. This ensures the token can benefit directly from every transaction on the network. As modules grow and generate more revenue, additional KITE is locked into their liquidity pools.

Non-Inflationary Economics. Unlike traditional PoS networks that rely on perpetual token inflation, KITE transitions rapidly from emissions-based rewards to a sustainable model powered entirely by protocol revenues. Initial emissions from a dedicated reward pool bootstrap early network participation, but the system is designed to transition to revenue-driven rewards funded by real AI service usage. This ensures token holders are never diluted by inflation–instead, rewards come directly from value created within the ecosystem.

Together, these mechanisms create a tokenomic system where real AI demand directly drives token value through multiple reinforcing loops: service usage generates revenues, successful modules lock more liquidity, and high-value participants are incentivized to continue creating value capture for the Kite network. The result is a token whose value is fundamentally linked to the utility and adoption of the Kite AI network.