Trustless Infrastructure for Autonomous AI Agents:

A Crypto-Native Architecture Integrating ERC-8004, x402, Chainlink CRE, and Space and Time

Whitepaper v1.0 — March 2026

Prepared for the 0xGasless Agent Infrastructure Research Initiative

ABSTRACT

Autonomous AI agents operating across blockchain networks face a fundamental trust trilemma: they must prove ***who they are*** *(identity),* ***what they did*** *(computation verification), and that the* ***data they relied on was correct*** *(data integrity). No single protocol addresses all three. This paper presents a unified, crypto-native architecture that combines four complementary protocols: ERC-8004 for on-chain agent identity, reputation, and validation; x402 for HTTP-native stablecoin payments; Chainlink's Runtime Environment (CRE) for*

Byzantine Fault Tolerant consensus-verified execution; and Space and Time (SXT) for ZK-proven SQL queries over tamperproof data. We provide a detailed technical analysis of each component, demonstrate how they compose into an end-to-end trustless stack, and identify the specific security guarantees each layer contributes—particularly in the context of AI agent wallet management, where we show that CRE's distributed signing model addresses critical vulnerabilities that ERC-4337 smart accounts alone cannot prevent. The resulting architecture eliminates every unverified trust assumption from raw blockchain data through agent decision-making to on-chain settlement, creating what we characterize as "AWS for AI Agents, but crypto-native"—where the trust model shifts from "trust a company" to "trust mathematics."

Keywords: *AI agents, ERC-8004, x402, Chainlink CRE, Space and Time, zero-knowledge proofs, account abstraction, trustless infrastructure, decentralized oracle networks, agent-to-agent payments*

I. Introduction

The emergence of autonomous AI agents—software entities capable of independent decision-making, API consumption, and financial transactions—represents a paradigm shift in computing. These agents increasingly operate across blockchain networks, managing digital assets, executing trades, and interacting with decentralized protocols. However, their proliferation exposes a fundamental infrastructure gap: no unified, trustless system exists for agent identity verification, computation validation, data integrity assurance, and payment authorization.

Current approaches rely on fragmented solutions. AI agents hold private keys on centralized servers, creating single points of failure. Reputation systems depend on unverifiable off-chain indexers. Payment authorization occurs through proprietary APIs requiring bilateral trust agreements. Historical data queries run against mutable databases with no cryptographic integrity guarantees. Each of these represents an unverified trust assumption that, if exploited, can result in catastrophic financial loss.

This paper presents an integrated architecture that eliminates every such assumption through the composition of four protocols, each addressing a distinct trust dimension:

ERC-8004 (Trustless Agents) — A set of three on-chain registries providing portable agent identity (ERC-721 NFTs), standardized reputation feedback, and pluggable validation mechanisms including stake-secured re-execution, zero-knowledge machine learning (zkML), and Trusted Execution Environment (TEE) attestations.

x402 (HTTP Payment Required) — An open payment protocol built on HTTP status code 402 that enables instant stablecoin micropayments between agents and services without API keys, KYC, or subscription agreements.

Chainlink CRE (Runtime Environment) — A programmable

execution layer where arbitrary TypeScript/Go workflows compile to WebAssembly and execute across Decentralized Oracle Network (DON) nodes with Byzantine Fault Tolerant consensus on every operation.

Space and Time (SXT Chain) — A verifiable database layer where SQL queries over indexed blockchain data produce zero-knowledge proofs (Proof of SQL), cryptographically guaranteeing both query correctness and underlying data tamperproofness.

We demonstrate that these four components, when properly composed, create a continuous chain of cryptographic verification from raw blockchain events through agent decision-making to on-chain settlement—achieving what we term a "fully trustless agent infrastructure stack."

II. Background and Related Work

A. The Agent Trust Trilemma

Autonomous agents operating in adversarial environments must simultaneously satisfy three trust requirements that existing infrastructure addresses only in isolation:

Identity Trust: How does a counterparty verify that an agent is who it claims to be, possesses the capabilities it advertises, and has a history of honest behavior? Traditional approaches use centralized registries (e.g., API key providers) or platform-specific reputation systems, both requiring trust in a single operator.

Computation Trust: How does a principal verify that an agent executed its task correctly—that it called the right APIs, processed the data accurately, and applied the correct business logic? Current solutions rely on server logs from the agent's operator, which constitute self-reported evidence.

Data Trust: How does any participant verify that the historical data underlying an agent's decision was accurate and untampered? Off-chain indexers (e.g., The Graph subgraphs) transform blockchain events into queryable databases, but the indexing process itself introduces an unverifiable trust assumption.

B. ERC-4337 and Account Abstraction

ERC-4337 introduced programmable smart accounts to Ethereum without consensus-layer changes. It replaces the traditional externally-owned account (EOA) model with a UserOperation flow: users construct and sign operations off-chain, bundlers submit them to an on-chain EntryPoint contract, which calls validateUserOp() on the smart account to verify signatures and enforce policies (spending limits, session keys, multisig) before executing the operation.

For human users, this is transformative—enabling gasless transactions, social recovery, and custom authentication. However, for AI agents, ERC-4337's security model has a critical blind spot: it validates the authorization (is the signature valid and within policy?) but not the intent (should this transaction exist at all?). The decision to create and sign a UserOperation still occurs on a single off-chain server. If the AI is compromised, hallucinating, or prompt-injected, it will produce perfectly valid UserOperations that pass all on-chain checks while executing adversarial actions. We analyze this gap in detail in Section VI.

C. The Chainlink Oracle Evolution

Chainlink's infrastructure evolved through several generations: price feeds (read-only data delivery), CCIP (cross-chain messaging), Functions (arbitrary off-chain computation), Automation (keeper-style triggers), and VRF (verifiable randomness). Each solved a specific slice of the oracle problem but required developers to manually integrate and orchestrate them. CRE represents a fundamental re-architecture: breaking the node software into modular capabilities (HTTP, EVM read/write, cron, consensus) that developers compose into workflows executing on specialized DONs with BFT consensus at every step.

D. Zero-Knowledge Proofs for Data Integrity

Zero-knowledge proof systems allow a prover to convince a verifier that a statement is true without revealing the underlying data. While zkVMs (e.g., RISC

Zero, SP1) can prove arbitrary computation, they are not optimized for database queries over large datasets. Space and Time's Proof of SQL is purpose-built for SQL analytics, achieving sub-second proof generation over million-row tables—an order of magnitude faster than general-purpose zkVMs for data processing workloads.

III. ERC-8004: On-Chain Agent Identity and Trust

A. Identity Registry

The Identity Registry implements ERC-721 with a URIStorage extension, minting each agent as a non-fungible token. The token's agentURI resolves to a JSON registration file (hosted on IPFS, HTTPS, or base64 data URIs) that declares the agent's name, description, service endpoints, supported trust models, and x402 payment support. Each agent receives a globally unique identifier in CAIP-10 format: eip155: {chainId}:{registryAddress}:{tokenId}.

The registration file supports arbitrary service endpoint declarations including A2A (Agent-to-Agent protocol), MCP (Model Context Protocol), ENS domain names, DIDs (Decentralized Identifiers), OASF skill taxonomies, and email contacts. An optional domain verification mechanism allows agents to prove control of HTTPS endpoints by publishing a .well-known/agent-registration.json file.

The reserved agentWallet metadata key requires EIP-712 signature verification for EOAs or ERC-1271 for smart contract wallets to change, preventing unauthorized redirection of an agent's payment address. When ownership transfers via the NFT, agentWallet automatically resets to the zero address and must be re-verified by the new owner.

B. Reputation Registry

The Reputation Registry provides a standardized interface for posting and querying feedback signals. Clients submit scores (a signed fixed-point int128 with configurable decimal precision) along with optional tags for on-chain filtering, endpoint URIs, and off-chain evidence references. Critically, the specification requires feedback authorization—before accepting a task, the

server agent signs a cryptographic authorization (EIP-191 or ERC-1271) allowing the specific client address to submit feedback. This pre-authorization includes expiration timestamps and index limits to prevent replay attacks while enabling batch authorization for watch-tower use cases.

The anti-spam design acknowledges that Sybil attacks remain possible but makes all reputation signals public and standardized, enabling competitive reputation aggregation services to develop sophisticated filtering, weighting, and spam detection algorithms. The on-chain getSummary() function returns aggregate feedback count and average score, enabling other contracts to gate access or adjust pricing based on agent reputation.

A particularly powerful integration pattern involves embedding x402 payment proofs within feedback evidence files. When a client includes proof that they actually paid for and consumed an agent's service, the feedback carries economic weight—creating what we term "economically-backed reputation signals."

C. Validation Registry

The Validation Registry provides generic hooks for requesting and recording independent third-party verification. Agent owners call validationRequest() specifying a validator address and a URI to the task data. The designated validator calls validationResponse() with a score (0–100, supporting binary pass/fail or spectrum outcomes) and optional evidence URI and hash.

The standard is deliberately agnostic about validation methodology, supporting: (1) stake-secured re-execution where validators re-run the agent's task and compare outputs; (2) zkML proofs verifying that a machine learning model produced a specific output for a given input; (3) TEE attestations from Trusted Execution Environments (e.g., Oasis ROFL, Intel TDX) proving code integrity; and (4) trusted human judges for subjective quality assessment. Multiple validationResponse() calls are permitted for the same request, enabling progressive validation states (soft finality followed by hard finality).

IV. x402: HTTP-Native Agent Payments

A. Protocol Mechanics

The x402 protocol, created by Coinbase, repurposes HTTP status code 402 (Payment Required) as a native payment rail. The payment flow operates in seven steps: (1) the agent issues a standard GET request to a resource; (2) the server responds with 402 and base64-encoded payment details (amount, token, chain, recipient) in an HTTP header; (3) the agent's wallet signs a USDC transfer authorization locally using EIP-3009 (transferWithAuthorization); (4) the agent retries the request with the signed payment proof in an X-PAYMENT header; (5) the server forwards the payment to a facilitator; (6) the facilitator verifies the signature and settles the transfer on-chain, paying gas on behalf of the agent; (7) the server delivers the resource along with a settlement receipt containing the transaction hash.

The agent never submits a transaction to the blockchain directly—the facilitator handles all gas costs and on-chain settlement. Each payment signature includes a unique nonce preventing replay attacks. Once settled and confirmed on-chain, transfers are final with no chargebacks, unlike traditional payment rails.

B. Facilitator Architecture

Facilitators are the trust-critical intermediary in the x402 flow. 0xGasless operates a facilitator at x402.0xgasless.com for the Avalanche chain, providing /verify and /settle endpoints. Multiple facilitators exist across the ecosystem (Coinbase, Cloudflare, AutoIncentive, Dexter, Stripe PayTo), each optimized for different networks. The agent trusts the facilitator to settle honestly—a trust assumption that CRE can eliminate by wrapping the settlement verification in a consensus-verified workflow, as detailed in Section VI.

C. Implications for Agentic Commerce

x402 fundamentally separates payment from identity. Traditional APIs require proof of identity (API keys, OAuth tokens) before accepting payment. x402 requires only proof of payment—any agent with USDC can access any x402-enabled

endpoint without prior registration, enabling truly permissionless agent-to-agent commerce. Combined with ERC-8004 identity, this creates a powerful separation of concerns: identity verification is optional but trust-enhancing, while payment is sufficient for access.

V. Chainlink CRE: Consensus-Verified Execution

A. Architecture and Execution Model

CRE (Chainlink Runtime Environment) is a programmable execution layer where developer-defined workflows compile to WebAssembly (WASM) and execute across Decentralized Oracle Network (DON) nodes. The architecture decomposes the Chainlink node software into modular capabilities—HTTP client, EVM client (read/write), cron triggers, confidential computing—each secured by independent capability DONs. A Workflow DON orchestrates the execution of the developer's callback function on every node in the network.

The TypeScript SDK (@chainlink/cre-sdk) exposes two runtime modes. Runtime (DON Mode) represents the DON's execution context for operations with automatic BFT guarantees—EVM writes, secret access, and aggregated results. NodeRuntime (Node Mode) represents individual node execution contexts for operations requiring independent evaluation —HTTP fetches to non-deterministic APIs, custom computation. The developer provides both the per-node function and the consensus aggregation algorithm (e.g., consensusMedianAggregation for numeric data, consensusIdenticalAggregation for deterministic results).

B. Consensus Computing Model

Every capability execution automatically includes consensus. When a workflow invokes a capability (fetching data from an API or reading from a blockchain), multiple independent nodes perform the operation. Their results are cryptographically verified and aggregated via a BFT consensus protocol, guaranteeing a single, correct, and consistent outcome. This means the entire

workflow—not just the on-chain parts—benefits from the same security and reliability guarantees as blockchain transactions.

For non-deterministic operations (e.g., fetching a random number from an API), the SDK provides runtime.runInNodeMode(), which follows a map-reduce pattern: each node independently executes the function (map), then the consensus algorithm reduces all individual results into a single trusted outcome. This pattern is fundamental to securely bringing off-chain data into the workflow.

C. Current Capabilities and Constraints

CRE is in Early Access with the following operational limits: 5 concurrent workflow executions per owner, 5 HTTP calls per execution, 10 EVM reads per execution, and a maximum 5-minute execution timeout. The TypeScript SDK compiles to WASM via Javy and QuickJS, inheriting WebAssembly's synchronous execution model (traditional async/await does not work; the SDK provides a .result() pattern instead).

These constraints are workable for agent payment authorization, reputation aggregation, and validation workflows but will constrain high-frequency trading agents. Cross-chain operations multiply EVM reads proportionally. The consensus mechanism introduces latency (seconds per capability call) that rules out sub-second operations but is acceptable for deliberate financial decisions.

D. Institutional Adoption

CRE is already being used by JPMorgan's Kinexys, Ondo, UBS Tokenize, and DigiFT for cross-chain settlement, tokenized fund redemption, and compliance-aware financial workflows. This institutional credibility is significant: the same infrastructure verifying billion-dollar institutional transactions can verify AI agent operations, providing a trust credential that purpose-built agent infrastructure lacks.

VI. CRE-Gated Wallets: Beyond ERC-4337

A. The Decision Boundary Problem

We introduce the concept of the decision boundary—the exact point in a transaction's lifecycle where the spending decision is made and who controls that point. In the raw EOA model, the AI agent holds the private key and signs transactions directly. The decision boundary is on a single server. In the ERC-4337 model, the AI constructs and signs a UserOperation off-chain, then the smart account's validateUserOp() enforces on-chain rules. The decision boundary remains on a single server—the on-chain validation only checks authorization constraints, not intent correctness.

In the CRE-gated model, the AI agent never holds a signing key. It submits structured intents to a CRE workflow endpoint. The DON evaluates the intent against both on-chain state (via EVMClient) and off-chain conditions (via HTTPClient), with every check independently performed by N nodes. Only after BFT consensus does the DON invoke its signing capability. The decision boundary is distributed across N independent nodes.

B. Attack Vector Analysis

We analyze five AI-specific attack scenarios across three wallet models:

Prompt Injection: An attacker manipulates the AI into sending funds to an adversarial address. In the EOA model, funds are drained instantly. In ERC-4337, the UserOperation passes if the amount is within policy and the signature is valid—the smart account cannot distinguish a manipulated intent from a legitimate one. In CRE-gated wallets, the DON nodes independently check the recipient against an approved list; the attacker's address fails the check across N nodes, blocking the payment.

Hallucinated Invoice: The AI fabricates a non-existent invoice and creates a valid payment. ERC-4337 passes because the UserOperation is structurally correct. CRE blocks it because each DON node independently fetches the invoice from the vendor API and confirms it does not exist.

Server Compromise: An attacker gains root access to the AI's server. In EOA and ERC-4337 models, the attacker has the signing key and can create valid transactions. In CRE-gated wallets, the signing key exists only within the DON; the attacker can submit intents but every intent still passes through N independent policy checks.

Spoofed Off-Chain Data: An attacker spoofs a delivery confirmation API. ERC-4337 cannot check external APIs at all. CRE's N nodes each independently call the API; if the attacker spoofs only some nodes' responses (e.g., via targeted DNS poisoning), consensusIdenticalAggregation detects the disagreement and fails safe.

Complex Business Rule Violation: A transaction satisfies on-chain spending limits but violates an off-chain business rule (e.g., audit freeze period). ERC-4337 can only enforce on-chain constraints. CRE checks the compliance API as part of the workflow, consensus-verified across all nodes.

C. Complementary Security Layers

CRE does not replace ERC-4337—it completes it. The two form a defense-in-depth architecture:

Layer 1 (CRE)—Intent Validation: Runs before any transaction is created. Verifies off-chain conditions, cross-source data consistency, business rules, and recipient legitimacy. Catches: hallucinations, prompt injection, spoofed data, AI compromise, and business rule violations.

Layer 2 (ERC-4337)—Authorization Validation: Runs on-chain after CRE signs the UserOp. Verifies signatures, nonces, spending limits, and gas policies. Catches: replay attacks, nonce manipulation, gas griefing, spending limit breaches, and unauthorized signers.

Neither layer alone is sufficient for AI agents. CRE without ERC-4337 loses replay protection and gas abstraction. ERC-4337 without CRE leaves the AI as a single point of failure controlling the signing key.

VII. Space and Time: ZK-Proven Data Layer

A. The Data Integrity Gap

Every layer in the architecture described above queries data: CRE workflows check spending histories, ERC-8004 reputation aggregation requires historical feedback analysis, validation decisions need cross-agent comparisons. Currently, these queries run against off-chain indexers that transform blockchain events into queryable databases—but the indexing process itself is unverifiable. A manipulated indexer could alter an agent's spending history, inflate reputation scores, or hide failed validations.

Space and Time (SXT Chain) addresses this gap as the first verifiable database layer for the EVM. Its architecture separates heavy off-chain computation from lightweight on-chain verification through three node types: Indexers that transform raw blockchain data (Ethereum, Bitcoin, Base, ZKsync, Avalanche) into structured relational tables; Validators that reach BFT consensus on cryptographic commitments (Dory, HyperKZG) representing the current state of the data; and Provers that execute SQL queries off-chain and generate zero-knowledge proofs attesting to both computation correctness and data integrity.

B. Proof of SQL

Proof of SQL is a purpose-built ZK prover for SQL analytics, capable of executing queries over 1M+ rows in under a second. Unlike general-purpose zkVMs, it is optimized for relational operations (JOIN, GROUP BY, WHERE, aggregation), achieving an order of magnitude faster performance for data processing workloads. The proof uses cryptographic commitment schemes (Dory and HyperKZG) that allow the on-chain verifier to confirm query correctness against a commitment signed by a quorum of staked validators, without ever seeing the raw data.

The commitment is updated every time new data is inserted into a table. Validators generate a threshold signature on each commitment update, tying the integrity of query results directly to economic risk (staked SXT tokens subject to slashing). This

creates a cryptographic root of trust referenced by every subsequent SQL query and ZK proof.

C. Integration with the Agent Stack

SXT integrates into the agent architecture at five specific points:

(1) ZK-Proven Reputation Aggregation: Instead of querying an unverifiable indexer for agent feedback histories, the agent-sdk queries SXT with SQL (e.g., SELECT agentId, AVG(score) FROM erc8004_reputation.new_feedback WHERE agentId = 42 GROUP BY agentId) and receives a ZK-proven result that mathematically guarantees the score reflects untampered on-chain feedback data.

(2) Verifiable Spending History: CRE wallet policy workflows query SXT for an agent's complete USDC transfer history (e.g., SELECT SUM(amount) FROM base.usdc_transfers WHERE from_address = '0xAgent' AND block_timestamp > NOW() - INTERVAL '24 hours') with a ZK proof anchoring the spending total to actual on-chain events rather than an estimated counter.

(3) Agent Discovery with Integrity: Complex cross-table queries for finding agents by capability, reputation, and activity (e.g., JOIN identity and reputation tables with filter conditions) produce results with cryptographic proof against manipulated leaderboards or falsified discovery rankings.

(4) Auditable Payment Analytics: All x402 settlement events indexed into SXT provide ZK-proven revenue reports, payment verification, and cross-agent billing reconciliation—critical for enterprise audit requirements.

(5) Validation Evidence Queries: Historical validation patterns (e.g., "has this agent been validated by at least 3 independent validators with scores above 80 in the last month?") return ZK-proven answers rather than unverifiable indexer results.

D. SXT and Chainlink Integration

SXT Chain is natively integrated with the Chainlink Network. This integration provides developers an efficient, consensus-based method to verify ZK query results, including native support for Chainlink

Functions to deliver ZK-proven query results from SXT directly to smart contracts. CRE workflows can query SXT and verify ZK proofs as part of their execution pipeline, creating a particularly powerful composition: CRE verifies computation correctness while SXT verifies data correctness.

VIII. Integrated Architecture

A. The Trust Chain

The complete architecture creates a continuous chain of cryptographic verification across six layers, each answering a distinct trust question:

ERC-8004 answers: "Who is this agent?" On-chain identity as ERC-721 NFTs, portable across chains, with standardized reputation and validation registries.

x402 answers: "How do agents pay each other?" HTTP-native stablecoin transfers requiring no API keys, KYC, or subscription agreements.

CRE answers: "Was the computation correct?" N independent DON nodes executing the same workflow with BFT consensus on every operation—HTTP calls, EVM reads, business logic.

SXT answers: "Is the data trustworthy?" ZK proofs over tamperproof indexed tables, anchored to validator-signed cryptographic commitments with economic security.

ERC-4337 answers: "Is this transaction authorized?" On-chain signature verification, nonce management, spending limits, and gas policy enforcement.

The EVM answers: "Is this state final?" Blockchain consensus on immutable settlement.

B. End-to-End Flow Example

Consider an AI agent that needs to pay $200 USDC for a data analytics API via x402. The complete flow through the integrated stack:

Step 1—Intent: The AI agent hits the API endpoint, receives HTTP 402 with payment details, and submits a structured payment intent (not a signed transaction) to its CRE workflow endpoint.

Step 2—Data Verification (SXT): The CRE workflow queries SXT for the

agent's 24-hour spending total. SXT's Prover executes the SQL query against indexed USDC transfer events and returns both the result ($347.50) and a ZK proof. The proof is verified against the on-chain cryptographic commitment.

Step 3—Policy Evaluation (CRE): Each of N DON nodes independently verifies: (a) the ZK-proven spending total plus $200 is within the $500 daily limit; (b) the recipient is on the approved vendor list (on-chain EVMClient read); (c) the recipient passes KYC via confidential HTTP; (d) the agent's ERC-8004 reputation score exceeds the minimum threshold. All checks pass BFT consensus.

Step 4—Signing (CRE DON): Only after consensus does the DON sign the USDC transfer authorization (EIP-3009). The AI agent never held a signing key.

Step 5—Settlement (x402): The agent retries the API request with the signed payment proof. The facilitator verifies and settles the transfer on-chain.

Step 6—On-Chain Validation (ERC-4337): If routed through a smart account, validateUserOp() provides an additional layer of signature verification, nonce check, and hard spending limit enforcement.

Step 7—Reputation (ERC-8004): After the service is delivered, the vendor agent issues a feedbackAuth signature. The client agent submits feedback (score, tags, payment proof URI) to the Reputation Registry.

Every step in this flow is either cryptographically proven (SXT), consensus-verified (CRE), or enforced by smart contract logic (ERC-4337, ERC-8004). No step trusts a single server, API, or operator.

IX. The "AWS for AI Agents" Thesis

The analogy to Amazon Web Services captures the breadth of primitives—compute (CRE/Lambda), identity (ERC-8004/IAM), payments (x402/Stripe), storage (SXT/RDS), wallets (ERC-4337/KMS), orchestration (CCIP/EventBridge), and audit logs (on-chain events/CloudTrail). However, the fundamental difference is the trust model: AWS provides convenience with implicit trust in Amazon, while this stack provides verifiability with trust in mathematics.

For institutional adoption—banks, asset managers, regulated entities—this distinction is the entire product. A Fortune 500 company considering AI agent infrastructure faces a binary choice: "Our server says the agent did X" versus "A Byzantine-fault-tolerant network of independent nodes cryptographically verified that the agent's action met all required conditions." The second sentence is the one that closes enterprise deals, and it is only achievable with CRE in the stack.

Without CRE, the stack is a collection of useful tools with centralized trust assumptions at the execution layer. Without SXT, the data feeding those tools is unverifiable. Without ERC-8004, agents lack portable, standardized identity. Without x402, payments require bilateral trust agreements. The architecture is irreducible—removing any component reintroduces a trust gap that enterprises cannot accept.

X. Limitations and Future Work

Several limitations constrain the current architecture:

CRE Operational Limits: The 5-concurrent-execution and 5-HTTP-calls-per-execution caps require batching strategies for high-throughput agent systems. The 5-minute execution timeout prevents long-running validation workflows (e.g., multi-step negotiation protocols). These limits will relax as Chainlink scales the infrastructure.

ERC-8004 Specification Maturity: The standard is in Draft status with active discussion on the Ethereum Magicians forum. Contract interfaces may evolve, requiring SDK updates. The Validation Registry portion is under revision with the TEE community.

SXT Query Coverage: Proof of SQL does not yet support all SQL functions. The set of indexed blockchain tables is expanding but not comprehensive. Custom smart contract event indexing requires configuration.

Cross-Chain Latency: CRE consensus adds seconds of latency per capability call. Combined with cross-chain CCIP messaging, total round-trip times for multi-chain agent operations may reach 10–30 seconds—

acceptable for deliberate financial decisions but not for high-frequency trading.

Language Ecosystem: CRE's SDK is TypeScript/Go, while most AI frameworks are Python-based. Bridging occurs via HTTP triggers, adding a layer of indirection and latency. A Python SDK would significantly improve developer experience.

Sybil Resistance: ERC-8004's Reputation Registry mitigates spam through feedback authorization but does not prevent coordinated Sybil attacks. Applications should layer additional requirements (stake, payment proof, reviewer reputation weighting) on top of the raw registry data.

Future work should address: native Python CRE SDK development, expanded SXT query function coverage, formal verification of CRE workflow correctness properties, standardized ZK-proven reputation aggregation algorithms, and cross-chain agent identity resolution protocols.

XI. Conclusion

This paper has presented a comprehensive, crypto-native infrastructure architecture for trustless autonomous AI agents. By composing ERC-8004 (identity and trust), x402 (payments), Chainlink CRE (consensus-verified execution), and Space and Time (ZK-proven data), we achieve what no single protocol provides: end-to-end cryptographic verification from raw blockchain data through agent decision-making to on-chain settlement.

The critical architectural insight is that trust in AI agent systems must be addressed at every layer simultaneously. Identity without computation verification allows impersonation. Computation verification without data integrity allows decisions based on manipulated inputs. Payment authorization without intent validation allows well-formed but adversarial transactions. Only the full composition eliminates every unverified trust assumption.

The CRE-gated wallet model represents a paradigm shift in AI agent security: moving the decision boundary from a single server to a distributed network of independent verifiers. This directly addresses the fundamental vulnerability in ERC-4337's security model for AI agents—that on-chain authorization validation cannot catch off-

chain intent corruption. The resulting defense-in-depth architecture (CRE for intent validation, ERC-4337 for authorization validation) provides security guarantees unavailable from either component alone.

As AI agents increasingly manage significant financial assets and make consequential decisions, the infrastructure supporting them must match the trust requirements of the institutions deploying them. The architecture described in this paper provides a credible path toward that standard—not through novel cryptography, but through the careful composition of battle-tested protocols into a system where every claim is independently verifiable.

The future of autonomous AI agents is not "trust our server." It is "trust mathematics."

References

[1] M. De Rossi, D. Crapis, J. Ellis, E. Reppel, "ERC-8004: Trustless Agents," Ethereum Improvement Proposals, Draft, August 2025.

[2] V. Buterin, Y. Weiss, D. Tirosh, S. Nacson, A. Forshtat, K. Gazso, T. Hess, "ERC-4337: Account Abstraction Using Alt Mempool," Ethereum Improvement Proposals, September 2021.

[3] Chainlink, "Introducing Chainlink Runtime Environment (CRE)," Chainlink Blog, October 2024.

[4] Space and Time, "Proof of SQL: A ZK Coprocessor for Sub-Second SQL Queries," Space and Time Foundation, 2025.

[5] Coinbase, "Introducing x402: HTTP Payment Required Protocol," x402.org, 2025.

[6] 0xGasless, "Agent SDK: ERC-8004 and x402 Compliant SDK," GitHub Repository, github.com/0xgasless/agent-sdk, 2026.

[7] Chainlink, "CRE SDK Reference: Core TypeScript," Chainlink Documentation, docs.chain.link/cre/reference/sdk/core-ts, 2025.

[8] S. Nazarov, "The Chainlink Platform Upgrade: SmartCon 2024 Keynote," Chainlink, November 2024.

[9] Space and Time, "Indexed Ethereum Data (ZK-Proven)," SXT Chain Documentation, docs.spaceandtime.io, 2025.

[10] Trail of Bits, "Six Mistakes in ERC-4337 Smart Accounts," Trail of Bits Blog, March 2026.

[11] Agent0 Labs, "Agent0 TypeScript SDK v0.31," npm: @kolhapuresatyajeet/agent0-sdk, 2025.

[12] D. Crapis, "ERC-8004 Genesis Month: Mainnet Launch," Twitter/X, January 2026.

[13] Composable Security, "ERC-8004: A Practical Explainer for Trustless Agents," Smart Contract Audits Blog, October 2025.

[14] 0xGasless, "AgentKit: On-Chain Execution for AI Agents," Avalanche Builder Hub, build.avax.network/integrations/0xgasless, 2025.

[15] Space and Time, "Introducing the SXT Token," Space and Time Blog, spaceandtime.io/blog, 2025.