Zypher Network Whitepaper

Zypher Team
research@zypher.network

Ver 0.4.0 (April 2025)

Abstract

Zypher Network introduces a decentralized cryptographic trust layer purpose-built for autonomous AI agents. Through innovations such as Proof of Prompt, zkPrompt, zkInference, zkTLS, and the Zytron AI Chain, Zypher enables verifiable, private, and auditable AI interactions across a wide array of domains. The network leverages zero-knowledge proofs to ensure that AI agents behave consistently, securely, and without revealing sensitive data. Its modular infrastructure, including a decentralized prover network and multi-execution engine support, offers composability, scalability, and real-world interoperability for use cases ranging from decentralized finance and gaming to AI compliance and digital labor. Zypher Network represents a foundational shift in how trust, privacy, and accountability are engineered into the future of AI.

Contents

1. Background **1**
1.1 The Rise of Autonomous AI 1
1.2 The Need for Verifiable Trust 2
2. Zypher Network Protocols and Architecture **3**
2.1 Proof of Prompt 3
2.2 zkPrompt 4
2.3 zkInference 9
2.4 zkTLS (Zero-Knowledge TLS) 11
2.5 Zytron: The On-Chain AI Engine 12
2.6 Decentralized Prover Network 12
2.7 Composability and Integration 13
3. Use Cases **13**
3.1 Decentralized Autonomous Organizations (DAOs) 14
3.2 Web3 Gaming 14

CRYP2NOVA

Archived copy · cryp2nova.com

3.3 Decentralized Finance (DeFi) 15
3.4 AI Agents and Digital Workers 15
3.5 Real-World AI Compliance & Regulation 16
4. Zytron AI Chain **17**
4.1 Web3-Compatible Interface 17
4.2 Multi-Execution Engine Support 18
4.3 Zero Gas Transactions 18
4.4 Native Cross-Chain Bridge 19
4.5 Data sovereign rollup 20
5. Conclusion 28

1. Background

1.1 The Rise of Autonomous AI

The rapid evolution of artificial intelligence (AI), particularly large language models (LLMs) and autonomous agents, has introduced a paradigm shift in the way digital systems interact, process information, and make decisions. These systems are no longer passive tools awaiting human input; they increasingly act autonomously, executing complex logic, initiating transactions, and even influencing governance in decentralized autonomous organizations (DAOs).

AI agents are now powering mission-critical processes in domains ranging from finance and healthcare to logistics and legal arbitration. Their ability to process vast datasets, detect patterns, and adapt in real-time makes them ideal for environments that demand speed, precision, and adaptability. However, this evolution comes with new challenges. As these systems grow more sophisticated, they also become more opaque. The decision-making processes of modern AI models, especially deep neural networks, are often non-intuitive and difficult to audit.

Furthermore, these agents are being integrated into increasingly decentralized infrastructures. Web3 applications, blockchain-based protocols, and decentralized finance platforms are beginning to embed AI agents into their logic flows. Yet, this convergence of decentralized systems and autonomous intelligence introduces a fundamental tension: how can we ensure that the behavior of these agents remains aligned, consistent, and verifiable without relying on centralized intermediaries?

As AI agents become economic actors, governance participants, and protocol managers, the need to verify their inputs, outputs, and reasoning steps becomes a critical requirement - not just a technical aspiration, but a security imperative.

Archived copy · cryp2nova.com

1.2 The Need for Verifiable Trust

Traditional web infrastructure was not built for verifiable trust. In centralized systems, users must blindly rely on the integrity of AI services, assuming that prompts are processed faithfully, results are generated honestly, and no unseen manipulation has occurred. This reliance on black-box logic and opaque data handling creates significant risk - especially as AI agents are deployed in adversarial or high-stakes environments.

In response, the Web3 movement has proposed a new paradigm: trustlessness. Blockchains eliminate the need for central trust by using consensus algorithms and cryptographic proofs to verify transactions. However, this model does not seamlessly extend to AI systems. Unlike deterministic smart contracts, AI models are probabilistic, compute-intensive, and dynamic by design. This makes them inherently more difficult to verify on-chain or through standard cryptographic techniques.

Zypher Network emerges to bridge this gap. It introduces a decentralized trust layer designed specifically for AI. At the heart of Zypher's approach is the use of **zero-knowledge proofs (ZKPs)** - a powerful cryptographic technique that allows one party to prove knowledge of certain information without revealing the information itself. Applied to AI systems, ZKPs make it possible to verify that a prompt was delivered unaltered, that a model returned a consistent response, and that no tampering occurred during the process - all without exposing the content of the interaction.

By integrating innovations such as **zkTLS**, **zkPrompt**, and **Proof of Prompt**, Zypher enables developers, users, and protocols to build AI-powered applications with auditable, tamper-evident interactions. This is further augmented by a decentralized prover network and **Zytron**, an on-chain AI engine designed for scalable, privacy-preserving AI execution.

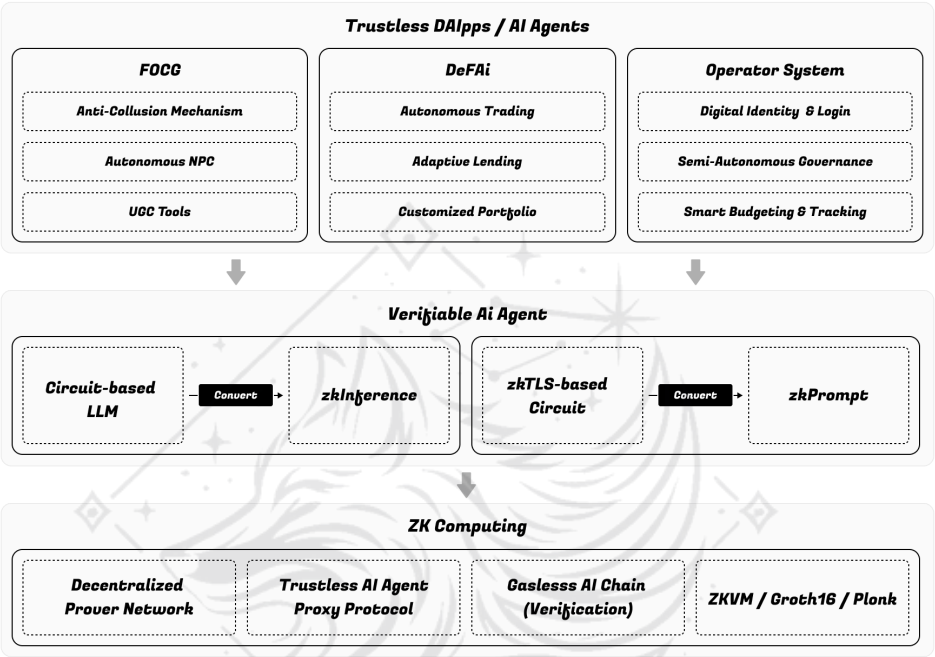
In a future where autonomous agents interact freely with on-chain contracts and off-chain services, the ability to verify those interactions without sacrificing privacy or scalability will be foundational. Zypher Network is building that foundation.

2. Zypher Network Protocols and Architecture

Zypher Network introduces a suite of cryptographic protocols and infrastructural components designed to bring verifiability, privacy, and integrity to the operation of autonomous AI agents. Each layer in the Zypher stack solves a different aspect of the trust problem in AI: verifying prompt

Archived copy · cryp2nova.com

integrity, ensuring encrypted communication consistency, enabling scalable proof generation, and anchoring these proofs into decentralized infrastructure.



2.1 Proof of Prompt

Proof of Prompt is Zypher's foundational protocol for ensuring the authenticity, consistency, and integrity of prompts sent to and received from AI models. It allows external parties to verify that an AI agent received a specific prompt and produced a corresponding output - without revealing the contents of the prompt or the response itself.

This is achieved using a hybrid cryptographic approach that combines:

- **Symmetric encryption** (e.g., AES, ChaCha20) for efficient data obfuscation,
- **Zero-knowledge proofs** (ZKPs) for non-revealing verifiability,
- And **commitment schemes** for immutability guarantees.

The result is a lightweight, privacy-preserving verification mechanism that can be applied to diverse use cases - from financial automation and DAO governance to

Archived copy · cryp2nova.com

Web3 gaming and identity management.

Proof of Prompt addresses the “black box” problem in AI by cryptographically guaranteeing that the behavior of the agent corresponds to a specific, untampered set of instructions. It lays the foundation for Zypher’s more advanced implementations.

2.2 zkPrompt

In large language models (LLMs), prompts are divided into system prompts and user prompts. Both types of prompts directly affect user experience and model performance, especially system prompts. A typical LLM interaction process is shown in Figure 1.

System prompts are provided by developers to define the context, tone, and boundaries of the LLM's responses. They serve as a guiding framework that shapes the AI’s behavior and style throughout the interaction. Mastering the design of system prompts has become a core competency for LLM developers, which is why most developers are reluctant to disclose them. Moreover, many LLMs operate as black boxes to users, making it difficult to ensure the consistency of system prompts.

System prompt consistency refers to the model consistently responding based on a predefined system prompt across different interactions, without any tampering, thus ensuring that its behavior aligns with expectations.

To address this, we propose **zkPrompt**, a method that uses zero-knowledge proof algorithms to verify the consistency of a model’s behavior - i.e., the consistency of its system prompt - without revealing the actual prompt. As shown in Figure 2, we compile the process of initializing the model with a system prompt into a circuit and incorporate a commitment module within the circuit to commit to the system prompt used. The zk circuit ensures that the committed prompt and the one used to initialize the model are the same. The commitment value is then output publicly by the circuit, allowing any user to verify the consistency based on the original commitment value.

Archived copy · cryp2nova.com

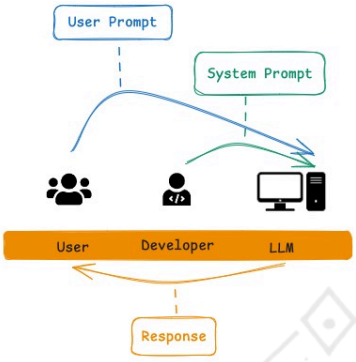


Figure 1

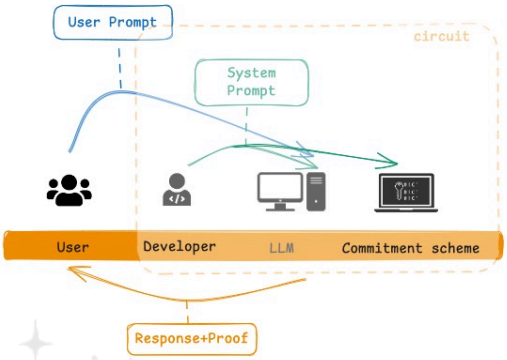


Figure 2

Cryptographic commitment is a protocol that allows one party to commit to a value without revealing the specific content of the value to another party, until the committer chooses to reveal it. Common commitment schemes include hash commitment and Pedersen commitment.

In **zkPrompt**, on one hand, the developer can keep the committed system prompt confidential; on the other hand, once the system prompt is committed, the developer cannot alter the committed content - this ensures the integrity of the system prompt.

Overall process:

1. The developer commits to the system prompt, then sends the commitment value on-chain
2. The circuit is initialized, generating a prover key and a verifier key, and the verifier key is sent on-chain
3. The user sends a User Prompt
4. The large model responds to the user's request, generating a Response + zk Proof, then sends it on-chain for verification
5. The smart contract uses the original system prompt commitment value to verify the zk proof; if verification passes, it proves the system prompt has not been tampered with

2.2.1 zkPrompt System Design

zkPrompt System Design

The zkPrompt protocol introduces a practical and efficient design for verifying large language model outputs using zero-knowledge proofs,

Archived copy · cryp2nova.com

without requiring users to directly engage with the LLM service. By structuring the system around four key participants - the User, Prover, Proxy, and LLM Provider - zkPrompt ensures that responses are authentic and untampered, even when delivered through potentially untrusted intermediaries. The design leverages cryptographic signatures and proof-generation mechanisms to validate that a Prover has faithfully relayed an LLM's output, while offering a privacy-preserving variant that enables verification without exposing prompt or response content. This architecture significantly lowers computational overhead compared to traditional ZKML frameworks and reflects a realistic trust model aligned with the reputational integrity of major LLM providers.

2.2.2 Participants

The zkPrompt protocol involves four actors:

- **User**
- **Prover**
- **Proxy**
- **LLM Provider (e.g., OpenAI, DeepSeek, etc.)**

2.2.3 Protocol Concept and Comparison

zkPrompt builds on ideas similar to zkTLS, embedding zero-knowledge proofs (ZKPs) into the interaction flow to ensure data authenticity and integrity. In this protocol, the user sends a request to the **Prover**, who forwards it via a **Proxy** to the **LLM provider**.

After processing, the LLM response is routed back through the Proxy to the Prover, and then finally to the user. The TLS session only involves the Prover, Proxy, and LLM provider - **not the user**, which creates a trust issue: the user can’t directly verify the response's origin or integrity.

To address this, **zkPrompt requires the Prover to produce a ZKP** proving that the response genuinely came from the LLM and has not been tampered

Archived copy · cryp2nova.com

with. This design significantly boosts the security and credibility of AI agent responses - crucial for high-stakes use cases like on-chain asset management, trading bots, and financial agents.

2.2.4 Design Advantages

Compared to frameworks like **ezkl**, zkPrompt is far more efficient. For instance, proving the output of even a basic GPT-2 model via ezkl could take hours - totally impractical for real-world applications.

Instead of verifying the full LLM execution, zkPrompt introduces a more **realistic trust model**:

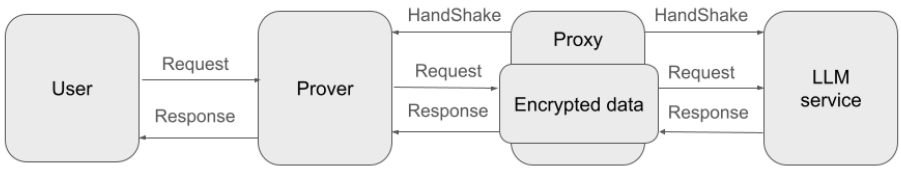
- Trust the **LLM provider's output**,
- Verify only that the **Prover hasn't altered the response**.

This mirrors real-world incentives - LLM providers have reputational and business incentives to act honestly, whereas Provers are more opaque and thus need stronger constraints.

2.2.5 Trust and Security Assumptions

It assumes no collusion between the Proxy and the Prover. For greater resilience, **Proxy decentralization** is encouraged in real-world deployments to prevent any single point of failure or abuse.

The current design also assumes that prompts and responses are posted on-chain in plaintext (with plans for privacy-preserving versions discussed below).



2.2.5 Technical Challenge & Core Solution

Archived copy · cryp2nova.com

A central challenge is: **How can we be sure that the encrypted response from the Prover is genuinely from the LLM and not forged locally?**

The solution: use a **Proxy signing mechanism**:

- When the Proxy receives the ciphertext from the LLM, it signs it and posts both the ciphertext and signature on-chain.
- The Prover then provides a ZKP proving that:
    - The decrypted ciphertext matches the response that includes the user's prompt.

This centers the proof around **correct symmetric decryption**, which can be efficiently implemented using techniques from papers like Dubhe.

2.2.6 Verification Steps

To verify that a response is authentic and untampered:

1. Confirm the **Proxy's signature** on the ciphertext.
2. Check that the **decrypted plaintext includes the user's prompt**.
3. Validate the **ZKP from the Prover** confirming correctness of decryption.

Once these steps are complete, anyone (e.g. miners, nodes) can confidently say the response is real and unaltered.

2.2.7 Privacy-Preserving Version

If users want to keep their prompt and response private:

- They post only the **hashes** of the prompt and response:

Archived copy · cryp2nova.com

- `H1 = Hash(prompt)`
- `H2 = Hash(response)`
- With the constraint: `response[i:j] == prompt`

The Proxy posts the encrypted ciphertext on-chain.

Then the Prover generates a ZKP asserting:

1. `Hash(witness) = H2` (response is correct)
2. `Hash(witness[i:j]) = H1` (prompt is included in the response)
3. `Decrypt(ciphertext) = witness`

This version allows verifiability **without revealing any sensitive content** - a powerful feature for regulated environments.

2.3 zkInference

AI Agents are being increasingly applied in the Web3 space, but currently, most AI Agents operate like “black boxes,” lacking transparency in their reasoning processes and making their behavior difficult to verify. This becomes especially problematic in multiplayer competitive games, where there is a risk of collusion between multiple AI Agents, as they are often controlled by the same entity. Such collusion may lead to coordinated behavior that undermines the fairness of the game.

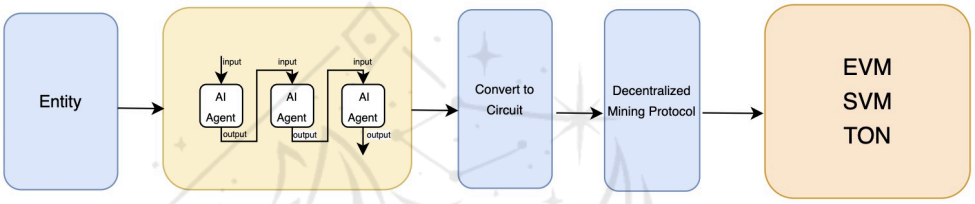
For example, imagine a game of Texas Hold’em poker where multiple AI Agents are controlled by a single entity - they could conspire against a specific player, severely affecting the gameplay experience and fairness. Therefore, the absence of a verifiable mechanism makes it difficult to regulate AI Agent behavior, posing a threat to the security of Web3 games.

To address this issue, we propose the **zkInference** framework. This framework uses zero-knowledge proof algorithms to ensure that AI Agents

Archived copy · cryp2nova.com

operate strictly according to predefined specifications or AI models, guaranteeing that their decision-making process adheres to principles of fairness, accuracy, and security.

In this way, the behavior of AI Agents can be verified without revealing the model or data, effectively preventing collusion and malicious actions among multiple AI Agents, and safeguarding the fairness and security of Web3 games.



Framework Features:

1. **Verifiability**: Leverages zero-knowledge proof technology to verify the behavior of AI Agents without exposing the model or data.

2. **Anti-collusion**: Effectively prevents collusion between different AI Agents, ensuring a fair gaming experience.

3. **Unlimited Compute Power**: Provides a decentralized mining marketplace, offering unlimited computational power for Verifiable AI Agents.

Aeroplane Chess is a game we have carefully crafted based on the zkInference framework, designed to provide a fair and transparent gaming environment. The core gameplay features one player competing against three AI Agents. However, in scenarios where AI Agent behavior is hard to verify, how can we ensure that these agents do not collaborate or are not controlled by the same entity?

To address this challenge, we innovatively combine game theory with the zkInference framework. Specifically, we use zk circuits to ensure that each AI Agent acts in the game with the goal of maximizing its own benefit, effectively preventing collusion among them.

Archived copy · cryp2nova.com

From a game theory perspective, we adopt the classic **MinMax algorithm**. As a game tree search algorithm, MinMax is particularly suitable for solving zero-sum games. In zero-sum games, one player's gain is exactly another's loss, with the total payoff remaining zero.

Applying the MinMax algorithm to the Aeroplane Chess game means that each player seeks to maximize their chances of winning. The MinMax algorithm recursively builds a game tree, exploring all possible moves. At each node in the tree, the algorithm evaluates the current game state and selects the optimal move based on whether the node is a maximizer or minimizer.

In this way, our Aeroplane Chess game not only ensures the independence and fairness of AI Agent behavior, but also enhances the strategic depth and competitiveness of the gameplay through the MinMax algorithm.

2.4 zkTLS (Zero-Knowledge TLS)

zkTLS is Zypher’s protocol for proving the contents and consistency of Transport Layer Security (TLS) sessions using zero-knowledge cryptography. Traditional TLS sessions are secure from eavesdropping but opaque to external verification. zkTLS changes that.

Zypher supports three deployment modes of zkTLS:

- **TEE-based**: Uses Trusted Execution Environments to securely capture and verify TLS session content. Offers high fidelity but requires specialized hardware.

- **MPC-based**: Employs Multi-Party Computation to jointly perform encrypted communications and disclose specific data selectively. This offers better decentralization but can be performance-intensive.

- **Proxy-based**: Sits as a man-in-the-middle that forwards and verifies TLS traffic. While less decentralized than MPC, it is significantly more performant and hardware-light, making it ideal for most

Archived copy · cryp2nova.com

real-world applications.

Zypher's implementation of zkPrompt uses the **proxy-based** zkTLS model to ensure that prompts and responses between the user and model remain verifiably consistent across encrypted sessions.

2.5 Zytron: The On-Chain AI Engine

Zytron is Zypher Network's on-chain AI execution layer. It acts as a decentralized coordination and verification engine for AI-related computations and proofs.

Key features of Zytron include:

- **Sharded P2P architecture** using the **Kademlia DHT algorithm** for efficient routing and load balancing.
- **Proof orchestration** across nodes, enabling distributed coordination of zkPrompt and zkTLS verifications.
- **Task allocation** based on node proximity and historical performance, optimizing for latency and trust.
- **On-chain anchoring**, allowing verified interactions and AI decisions to be recorded in an immutable, decentralized ledger.

Zytron is the backbone of Zypher's verifiable AI system. It enables not just proof generation, but scalable coordination and enforcement of trust policies across AI agents operating in the network.

2.6 Decentralized Prover Network

The **Decentralized Prover Network** enables scalable generation and validation of zero-knowledge proofs. Proofs generated by zkPrompt, zkTLS,

Archived copy · cryp2nova.com

or other components are computationally heavy, and running them through a centralized server would defeat the trustless nature of the system.

Zypher's prover network solves this by:

- Allowing any qualified node to contribute computing power toward proof generation.
- Implementing a reputation and staking model to discourage dishonest behavior.
- Coordinating provers through Zytron to assign tasks efficiently and verify output correctness.

This network is essential for Zypher's scalability and decentralization goals. It ensures that verifiability does not come at the cost of speed or accessibility, enabling AI verification to operate at internet scale.

2.7 Composability and Integration

Zypher's modular protocol stack is designed for easy integration across multiple domains. Developers can compose Zypher's components into their own products and pipelines in various ways:

- Embed zkPrompt into API gateways for LLM-based apps.
- Use zkTLS as an integrity layer for privacy-preserving AI assistants.
- Plug Proof of Prompt into Web3 DAOs to verify AI-powered governance proposals.
- Anchor AI interactions into on-chain smart contracts via Zytron.

This composability ensures Zypher can serve as both a foundational protocol layer and a developer-facing toolset for building trustworthy AI applications.

Archived copy · cryp2nova.com

3. Use Cases

Zypher Network's trust infrastructure is highly versatile, enabling secure and verifiable AI across a wide range of industries and applications. Below are several prominent use cases where Zypher's protocols add immediate, mission-critical value:

3.1 Decentralized Autonomous Organizations (DAOs)

As DAOs become more sophisticated, many are integrating AI agents to automate governance, strategy, and operations. However, without verifiable trust layers, there's no guarantee that the AI outputs - proposals, votes, or resource allocations - are free from tampering or misaligned behavior.

Zypher enables DAOs to:

- Verify that AI-generated proposals are based on consistent and untampered prompts.
- Prove that governance automation agents are executing instructions faithfully.
- Log interactions transparently while preserving sensitive DAO strategies.

With Proof of Prompt and zkPrompt, DAOs can ensure their AI collaborators are trustworthy and accountable - without exposing private deliberations or voting logic.

3.2 Web3 Gaming

Archived copy · cryp2nova.com

AI plays a growing role in Web3 gaming - from procedural content generation to NPC behavior and narrative progression. When rewards and assets are on-chain, game fairness must be cryptographically enforced.

Zypher secures game logic by:

- Ensuring prompts to AI game engines remain consistent (no cheating or biased instructions).
- Verifying AI-generated content is based on predefined rules.
- Enabling players to challenge outcomes or decisions using ZK proofs.

zkPrompt can run invisibly in the background, acting as an integrity engine that guarantees provable fairness in games where AI plays a role in determining in-game economics or player outcomes.

3.3 Decentralized Finance (DeFi)

DeFi protocols are beginning to explore AI-powered tooling for risk modeling, fraud detection, and automated trading. However, the opaque nature of AI models introduces a new attack vector: unprovable behavior.

Zypher brings accountability to DeFi AI by:

- Verifying that bots and models execute trading or lending strategies from auditable prompts.
- Ensuring that pricing, liquidation, and collateral logic remains unaltered by outside interference.
- Enabling DeFi users to request and audit ZK-verified reports of AI decision chains.

Archived copy · cryp2nova.com

In DeFi, trust is capital. Zypher ensures that AI logic can be trusted just like smart contracts.

3.4 AI Agents and Digital Workers

AI agents acting on behalf of users - browsing the web, negotiating deals, managing tasks - need to be auditable and provable in their decisions. Without a trust layer, users are left in the dark about how and why agents act.

Zypher enables verifiable autonomy by:

- Proving that agents act according to pre-defined prompts and user instructions.
- Verifying interactions with external APIs or services using zkTLS.
- Auditing agent decisions after-the-fact with tamper-proof logs.

As agents evolve into intelligent "workers" in crypto ecosystems, Zypher ensures these workers remain transparent and trustworthy.

3.5 Real-World AI Compliance & Regulation

Governments, enterprises, and institutions are increasingly subject to AI-related regulations that demand auditability, transparency, and accountability. Zypher's trust infrastructure offers a compliance-friendly framework for verifiable AI.

Applications include:

- Healthcare AI needing to prove model usage without exposing patient data.
- Financial institutions validating that AI models adhere to regulatory guidelines.

Archived copy · cryp2nova.com

- AI service providers offering proof-of-behavior guarantees to clients.

With zero-knowledge proofs at the core, Zypher enables provable compliance without sacrificing user privacy or intellectual property.

4. Zytron AI Chain

Zytron is Zypher Network's high-performance execution layer purpose-built for scalable, low-latency, and verifiable AI computation. Designed from the ground up for decentralized AI workloads, Zytron introduces architectural innovations to support zk-proofs, composable AI agent logic, and flexible state execution - all while maintaining interoperability with the broader Web3 ecosystem.

4.1 Web3-Compatible Interface

To support a seamless developer experience, Zytron exposes a dual-interface system. While Zytron introduces numerous architectural optimizations that diverge from Ethereum's native interface - particularly around transaction latency, state addressing, and proof generation - it maintains compatibility through a dedicated Web3 layer.

This compatibility layer:

- Translates Zytron-native transactions and data formats into standard Web3 JSON-RPC equivalents.

- Handles address mapping, state diff interpretation, and pending-state queries.

- Automates Space Access List estimation and transaction normalization for EVM-based tooling.

This approach enables developers to deploy smart contracts and interact with Zytron using familiar Web3 tools like MetaMask, Hardhat, and Foundry,

Archived copy · cryp2nova.com

while still benefiting from Zytron's performance and verifiability enhancements.

4.2 Multi-Execution Engine Support

Zytron is designed to be execution-engine agnostic. While its architecture is optimized for AI and zk-based workloads, it supports multiple contract execution models in parallel. This includes:

- **EVM (Ethereum Virtual Machine)** for compatibility with Solidity-based contracts.
- **UTXO (Unspent Transaction Output)** for lightweight, parallelizable, and deterministic transaction flows.

Execution models can be assigned at the **address state group** level. For example:

- AI agent token accounts may operate under a UTXO model to avoid concurrency issues in distributed logic execution.
- High-complexity smart contracts may default to the EVM path for flexibility and compatibility.

Future extensions may introduce support for WASM-based engines or domain-specific VMs depending on evolving application requirements.

4.3 Zero Gas Transactions

Zytron natively supports **ERC-4337-style account abstraction**, allowing it to implement zero-gas transaction flows through delegated transaction sponsorship. By abstracting away fee payments from end-users, Zytron enables seamless onboarding and agent autonomy.

Key features include:

Archived copy · cryp2nova.com

- **Gasless onboarding** for new wallets and AI agents.
- **Transaction sponsorship** enforced at the sequencer level, allowing dApps or infrastructure providers to subsidize execution costs.
- **Custom fee logic** embedded in account abstraction logic, enabling novel incentive models and user experiences.

This feature is particularly important for AI agents and automated systems, which require frictionless interaction with the chain without managing native gas tokens.

4.4 Native Cross-Chain Bridge

Zytron's modular sharding architecture enables secure and efficient cross-chain asset flows. Each Zytron shard can be mapped to a dedicated **state group** corresponding to a smart contract or bridge address on an external blockchain (e.g., Ethereum, Linea, or other L2s).

Cross-chain bridging is supported via:

- **Deposit transactions** originating from a canonical bridge contract on the source chain.
- **Minting logic** within the corresponding Zytron shard to reflect locked tokens as native Zytron-wrapped assets.
- Optional **zero-knowledge verification** of source-chain state to further increase bridge security.

This model allows for efficient AI-agent interactions across multiple blockchains while keeping computation and verification localized within Zytron's optimized environment.

Archived copy · cryp2nova.com

4.5 Data sovereign rollup

4.5.1 Sequencer

The Sequencer is a critical component in Layer2 systems, acting as the first entry point that determines the order of transactions. It organizes transactions from the mempool and packages them into Layer2 blocks, which are then sequentially grouped into Batches. Subsequent transaction execution and processing are based on the block order generated by the Sequencer.

Transactions sent to Zytron by users are first sorted by the Sequencer, which establishes a global order determined by a decentralized sorter. This sorter handles only the ordering and basic signature verification of transactions, without executing the transactions themselves or running any smart contracts. Therefore, the Sequencer cannot package complete transaction receipts, but instead, it packages the ordered transactions into Batches that are finalized by Zytron's transaction sharding service.

The specific packaging process includes:

- Sequencer nodes utilize a non-P2P mempool, where transactions sent to a node are locally packaged. Sequencer nodes package transactions at regular intervals, forming a transaction block.
- Transactions within this block are structured into a Merkle tree, and the Merkle root is broadcast within the Sequencer's P2P network. This broadcast process involves only the Merkle root and not the full transaction body, further reducing data volume by aggregating signatures from the transaction block for verification.
- The transaction block data packaged by the Sequencer is then submitted to an external Data Availability (DA) layer. To prevent delays caused by lengthy confirmation cycles in the DA, the Sequencer temporarily submits data to a "weak" DA.
- The transaction blocks generated by the Sequencer are further packaged into Batches by the Sequencer's PBFT consensus and are finally submitted for execution in subsequent service shards.

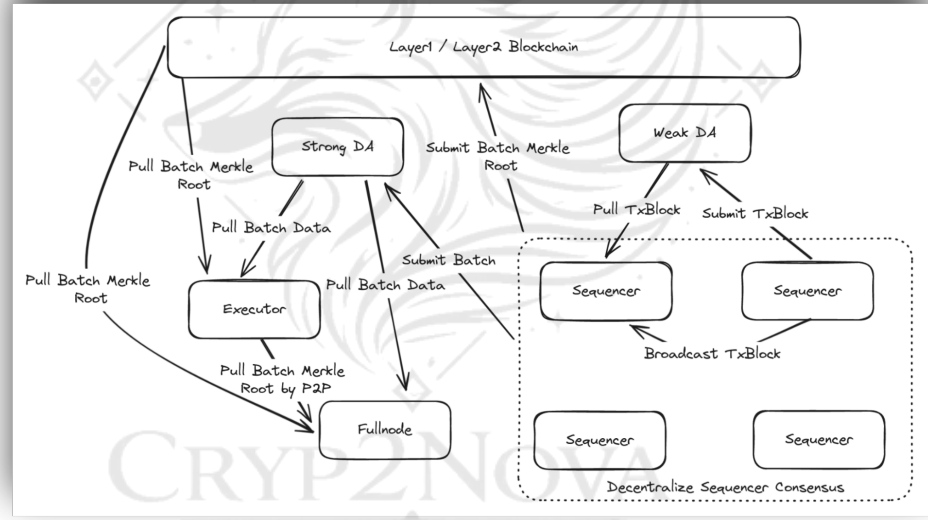
4.5.2 External DA

Zytron employs two types of DA layers: "strong" and "weak" DA. The "weak" DA is used during the Sequencer's transaction block packaging to record

Archived copy · cryp2nova.com

data. It does not require consensus support and serves merely to swiftly transmit specific transaction block information within the Sequencer network, utilizing publicly accessible servers like S3 and other centralized services. This weak DA primarily supports Zytron's "pipeline" style transaction execution optimization strategy.
The strong DA records data from the Sequencer-packaged Batches that serve as input for final contract execution. This data must be stored in a highly accessible system, emphasizing data accessibility and often leading to longer data submission cycles. The data in strong DA is more for verifying on-chain data outcomes at fixed points.

A diagram illustrating the differences between strong and weak DA is as follows:



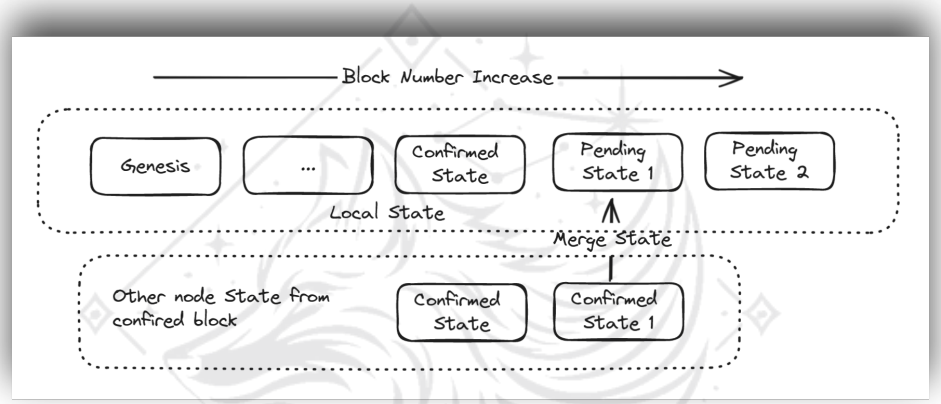
In the interaction process between the DA and the Sequencer, Sequencers broadcast data using a proactive push mode via P2P. This broadcast is mainly for quickly realizing data dissemination before consensus to reduce transaction processing delays. Specific transaction data is then actively pulled from the respective nodes by the Sequencer.

4.5.3 Sequencer Pending State

In game processes, there are often a large number of transactions sent to the network frequently. Due to the low latency requirements of games,

Archived copy · cryp2nova.com

requiring users to wait for the transaction results to be confirmed by the chain before returning them would significantly deteriorate the user's gaming experience. Therefore, when users send transactions to the Sequencer, the Sequencer maintains a local Pending State. Users can continuously pull the Pending state from the Sequencer transaction node they are connected to, and issue new transactions before the previous ones are finally confirmed. The Pending State recorded in the Sequencer is continuously synchronized with the final state on the chain to correct its own Pending State.

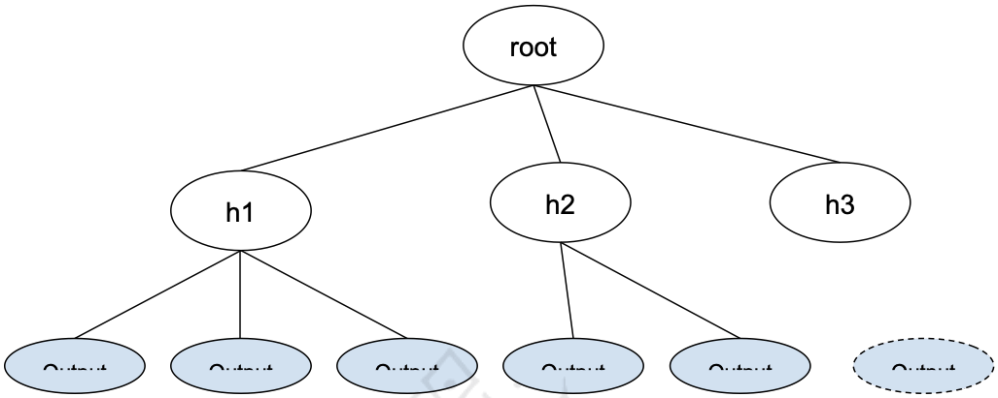


4.5.4 Assets zk rollup

In the context of Layer 3 (L3) in blockchain technology, assets differ significantly from data in that they have clear value attributes and require liquidity. To ensure the security of assets within Layer 3, which is built on the more secure foundational layers (Layer 1/Layer 2), we've designed a zk-rollup-based asset management solution tailored for use in the Zytron Kit issued at Layer 3.

Asset Management with zk-Rollups:

Archived copy · cryp2nova.com



Asset Transactions in L3:

- Merkle Tree Storage: All asset transactions generated within L3, along with deposit and withdrawal transactions linked to L1/L2 source chains, are stored in a Merkle tree. Each branch of this Merkle tree consists of three leaves, each representing an output.
- UTXO Model: By adopting the UTXO model's transfer algorithm, we facilitate the zk aggregation of all transaction validity checks, enabling immediate verification on L1/L2 without the need for a 7-day challenge period, unlike other Optimistic Rollups (OP).

Transaction Types and Circuits:

- Deposit: Exists on the source chain and does not require its signature to be incorporated into the circuit as it is a completely public input.
- Transfer: Involves the validity of transactions where both inputs and outputs are submitted, with user signatures embedded in the circuit to validate the transactions. A field-friendly signature algorithm is used for L3 transfer transactions to enable efficient proof within the circuit.
- Withdraw: Treated similarly to the transfer algorithm, except that the output's recipient is designated as an account format corresponding to the source chain, allowing participation in the Merkle tree proof and preventing its use by any other input.

Circuit-Driven Transaction Validity:

- Leaf Proofs and Merkle Root Updates: Changes in the leaves lead to updates in the Merkle root. The validity of these changes is proven within the circuit, reducing the on-chain computation required to update the Merkle root.

Archived copy · cryp2nova.com

- Public Inputs Submitted to the Source Chain: Includes deposits and associated outputs, transfers with related inputs and outputs, withdrawals with generated inputs and special outputs, and the final Merkle tree root, with all asset transaction validity proven through ZKPs.

UTXO Structure:

- Asset Types and Amounts: Both are of type u256, capable of handling both ERC20 token types and ERC721 NFT types.
- Contractual Mapping on Source Chain: By establishing appropriate mappings within source chain contracts, assets can be bridged to L3, with the security assumption of assets based on the source chain and reducing reliance on L3's security.
- The implementation plan includes creating diagrams that visually explain the UTXO transactions processed into ZK inputs and outputs, the transition from the old to the new Merkle root, and mapping tables that correlate asset types and amounts to token and NFT types, along with the mapping relationships. This structured approach enhances the understanding and transparency of asset management within Layer 3, leveraging the security features of the underlying blockchain layers.

4.5.5 Service Sharding

Zytron is designed to provide low-latency on-chain gaming services. Therefore, to achieve this, contracts running on Zytron are executed by sharding the data services of the chain nodes.

4.5.6 Kademlia Node Data Sharding

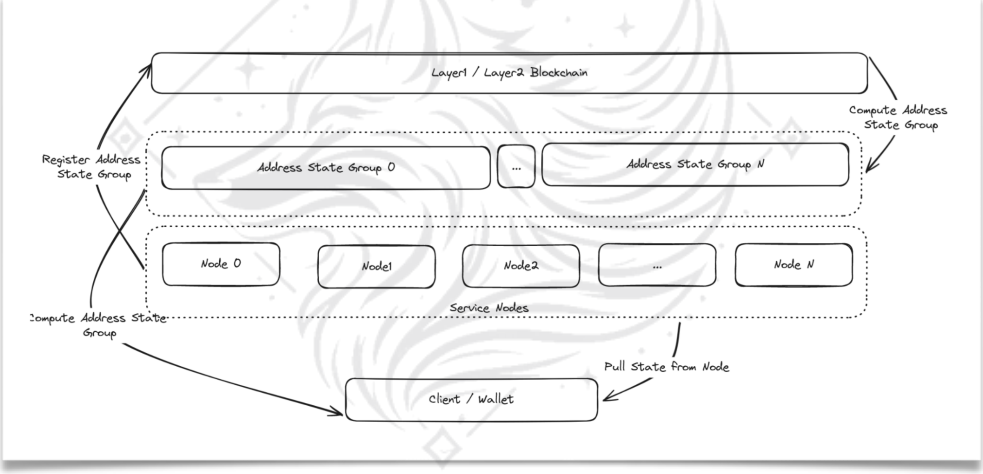
Kademlia is a structured hash-based DHT algorithm, using PeerId as Zytron connects nodes that specifically execute contract validation through a P2P network. Addressing between nodes is accomplished using the structured Kademlia algorithm. Additionally, the execution process of contracts in the network is sharded based on the Kademlia node distance, distributing the execution of specified contracts across Zytron’s network nodes according to Kademlia’s distance rules. To become an executing node in Zytron, the following process must be completed:

- Each executing node in Zytron can receive transaction blocks sent from the Sequencer and reads the Batch that has been

Archived copy · cryp2nova.com

consensus-approved from the upper-layer blockchain and specific Batch data from DA.

- Nodes generate a 160-bit node id using their private key.
- Each node must register on the upper-layer blockchain, and the blockchain's contract will record the current node's ID.
- After joining the network, a node initiates a query operation on itself through the P2P protocol to be updated in the routing tables of relevant neighbors.
- Initiates state data synchronization requests to surrounding nodes.
- After synchronization is complete, register the successful synchronization results (the Merkle Root of the data state recorded by the node) with the upper-layer blockchain.



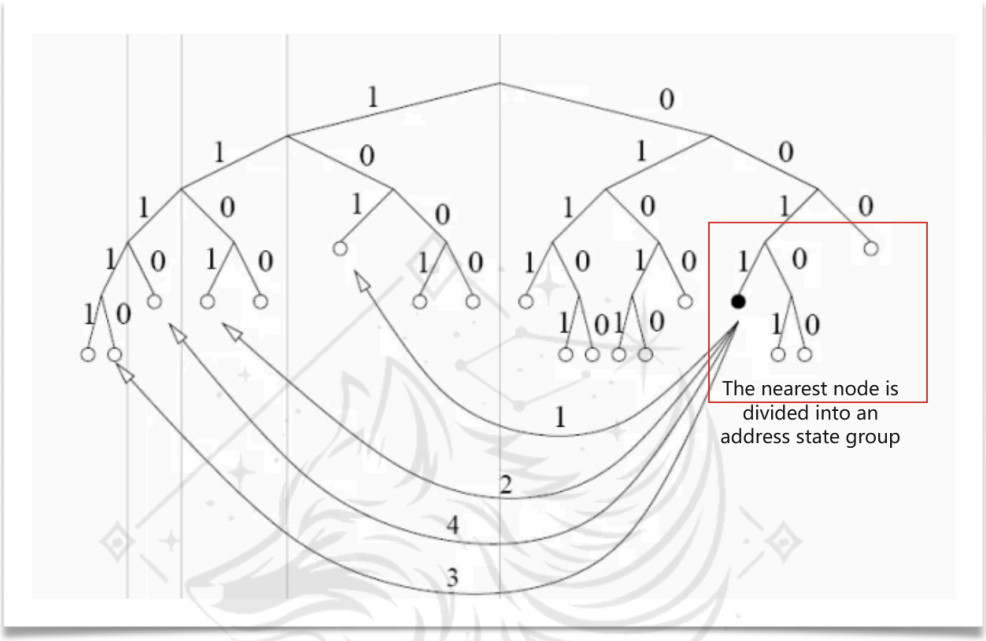
Once a node becomes an executing node, it initiates state data synchronization requests to nearby nodes, syncing relevant data such as account addresses closest to its node ID, balances, nonces, etc., into its storage.

4.5.7 Address State Group

Executing nodes in Zytron do not record the global state; each executing node only records storage information related to the addresses closest to it. To ensure that transactions can still be verified normally even when a node is offline, every time an executing node syncs storage data from an address close to it, it joins or creates an address state group. Each address state group consists of eight members, who locally update the address's state using the BFT algorithm.

Archived copy · cryp2nova.com

Initialization algorithm for address state groups:



When a transaction is sent to a Zytron node for execution, the address state group bases on the transaction's Address Space to combine multiple address state groups to execute smart contracts and update the state involved in this transaction, achieving an update to the transaction state. Additionally, since Zytron has features like Address Space Access List and Key Space Access List, the merging of address state groups can be done before contract execution, without waiting for the contract execution to merge.

When a node goes offline, the related address node group will exclude the node and record this exit on the upper-layer blockchain.

4.5.8 Transaction Space Access List

Since data in Zytron nodes is sharded, executing nodes do not hold a complete global state. Even though address state groups join together to execute transactions, it is challenging to dynamically keep adding connections and executing transactions. Therefore, to ensure data can be properly sharded for processing, when sending transactions to the chain, the transaction must include State Change information, known as Space Access

Archived copy · cryp2nova.com

Limit. This information includes changes to Balance, Nonce, and Storage. Upon receiving a transaction, nodes construct a state locally that meets the contract execution conditions within the transaction `to` address's address state group node. If during execution, the accessed data exceeds the range recorded in the Space Access List or differs from the state changes, the transaction is deemed to fail.
After each Batch execution, address state groups submit a State Root and related zkproof or Fraud Proof to the upper-layer chain.

4.5.9 Key Space Estimation

For most EVM-class or full-state contract execution engines, there is no Space Access List field by default, and no one can know the state of transaction execution before the smart contract execution ends. Therefore, to assist clients in properly marking this field, Zytron provides a Space Access Limit estimation feature. This feature sends a transaction, executes a smart contract based on a specified state, and constructs a Space Access Limit based on the data accessed and generated during the contract execution.

4.5.10 Custom Network

In current blockchain P2P networks, many projects are based on libraries like libp2p or their own custom libraries, which are developed specifically for blockchain applications. Their most crucial feature is the rapid propagation of blocks and transactions, hence they emphasize broadcast algorithms such as the gossip algorithm. These are used to ensure messages are broadcasted network-wide within a short time frame. However, gaming scenarios often involve limited node-to-node message transmission, sometimes between just two nodes, and require these connections to be stable and efficient, not arbitrarily disconnected, or to ensure quick recovery when they are disconnected. Blockchain P2P libraries do not inherently have these features. Therefore, we have undertaken customized development for our P2P network library.

Firstly, regarding the network transmission protocol, in high-frequency operational games, such as those running at 60 frames per second and updating 60 times a second, latency must be below 16ms to avoid dropped frames and stuttering. For this, we need not only to optimize the

Archived copy · cryp2nova.com

performance of the client but also the network transmission, which takes up most of the time. Traditional P2P networks generally use the TCP transmission protocol, but TCP includes reliable retransmissions and has congestion control mechanisms. If there is significant traffic on the network, this can extend TCP's transmission time, which does not meet the requirements for gaming. Therefore, we have modified a P2P network library based on the KCP transmission protocol, which is implemented over UDP. KCP is a fast and reliable protocol that can reduce average latency by 30%-40% and maximum latency by three times at the cost of 10%-20% more bandwidth than TCP. KCP is purely algorithmic and does not handle lower-level protocols (such as UDP); it requires users to define how lower-level data is sent, using a callback provided to KCP. Even the clock needs to be externally supplied, as there are no system calls made internally.

To ensure the stability of node connections, we have optimized the core addressing algorithm of the P2P network, adopting a stable P2P network structure based on a dual DHT. One layer uses the node's peer id, performing an XOR distance calculation under the first 160 bits using the Kademlia algorithm, and keeps the DHT's bucket at 8 to cope with network connections and fluctuations. Additionally, we use stable Kademlia to maintain the stability of the network nodes' table caused by entries and exits. The other layer uses a distance calculation based on socket IP addresses, using physical location proximity. The advantage is that if two nodes cannot connect directly, for example, both are within a LAN and restricted by Port-Restricted Cone or Symmetric, then when they cannot connect directly through NAT and hole punching, a relay algorithm is used. This involves using a third node to establish a stable connection, and this third node must not be too far from the other two to improve transmission efficiency and reduce latency.

5. Conclusion

As artificial intelligence continues to expand its influence across decentralized ecosystems, the need for provable trust mechanisms becomes paramount. Zypher Network rises to this challenge with a comprehensive stack of cryptographic protocols and decentralized infrastructure that transforms opaque AI interactions into transparent,

Archived copy · cryp2nova.com

auditable, and privacy-preserving operations. From DAOs to DeFi, gaming to enterprise compliance, Zypher's architecture ensures that AI agents can be held accountable without compromising performance or privacy. By embedding verifiability at the core of autonomous systems, Zypher lays the groundwork for a new era of trustworthy AI, one that is aligned with the principles of decentralization, user sovereignty, and cryptographic assurance. In doing so, Zypher is not just building a toolset, but defining the blueprint for the next generation of AI infrastructure.

Archived copy · cryp2nova.com