# A Decentralized Framework for Multi-Agent Systems Using Datachain Technology

Hugo SQR

**abstract**: The rapid advancement of artificial intelligence has led to an increasing need for effective multi-agent systems, yet current frameworks often rely on centralized communication methods through HTTP requests and APIs. This paper presents a novel decentralized framework for multi-agent communication built on datachain technology (Irys), enabling agents to interact in a public, trustless environment. The proposed architecture introduces three key components: Providers for data retrieval and context addition, Orchestrators for request management, and Workers for task execution. To ensure system reliability and security, we implement multiple safeguarding mechanisms including a reputation system, staking requirements, and a mandatory testing phase for new agents. The framework leverages a tagging system for efficient data organization and retrieval through GraphQL queries, while maintaining separation between data storage and request handling. A notable feature is the integration of dynamic NFTs for agent registration and metadata management, enabling real-time reputation tracking and status updates. Our implementation includes robust security measures such as multi-provider consensus for critical tasks and malicious actor detection. Initial testing demonstrates that this framework provides a reliable foundation for decentralized agent interactions while maintaining system integrity through various validation mechanisms. The architecture presents a significant step forward in enabling autonomous agent collaboration in decentralized environments

## Disclaimer

This paper presents a conceptual framework and theoretical vision for a decentralized multi-agent system, reflecting personal insights and potential approaches to addressing challenges in this domain. The mechanisms, processes, and implementations described herein are primarily theoretical constructs and should be understood as exploratory ideas rather than definitive solutions.

Not all concepts presented need to be implemented to create a functional system, and different projects may choose to prioritize certain aspects based on their specific requirements, constraints, and objectives. The choices made in implementing such systems often involve trade-offs, particularly in relation to the blockchain trilemma (decentralization, security, and scalability), and what works best for one project may not be optimal for another.

It is important to note that this paper is not intended to criticize or diminish the value of existing projects in the space. Many current implementations have made thoughtful and strategic decisions in their architectures to best serve their target use cases and user bases. The diversity of approaches in the ecosystem contributes to its richness and resilience, with different projects optimizing for different aspects of the trilemma based on their unique requirements and goals.

This framework should be viewed as a contribution to the ongoing dialogue about decentralized systems rather than a prescriptive solution, acknowledging that the field continues to evolve through innovation, experimentation, and collaborative development.

## 1. Introduction

The field of artificial intelligence has witnessed remarkable progress in recent years, particularly with the emergence of large language models (LLMs). Early AI agents showed remarkable innovation, with projects like AIXBT demonstrating the true potential of autonomous agents. What made AIXBT particularly impressive wasn't its ability to write tweets or perform analysis, but rather its system for real-time data retrieval and integration, highlighting how rich data contexts can enhance agent capabilities [1]. However, recent developments in the AI agent space have shown less innovation, with most current "AI agents" being merely wrappers around the GPT API, offering limited genuine innovation in terms of agent architecture and autonomous capabilities [2].

This limitation becomes particularly evident when examining the current landscape of agent frameworks. The market has essentially split into two distinct categories. On one side, we have user-friendly platforms similar to Pump.fun but dedicated to agents (such as Virtuals Protocol and vvaifu), which democratize agent creation by allowing users to launch agents and associated tokens with minimal technical expertise. While these platforms have successfully lowered the barrier to entry, they often sacrifice sophistication for accessibility. On the other side, we find more technically advanced frameworks attempting to create sophisticated agents, but these typically require significant coding expertise and, more importantly, still rely heavily on HTTP/API calls for core functionality.

The need for open-source frameworks, as demonstrated by projects like ElizaOS, has become increasingly apparent. Open-source solutions not only foster collaboration and innovation but also enable the development of truly decentralized and autonomous agents. Such frameworks are essential for moving beyond the current paradigm where agents are essentially API-dependent entities [3].

At the heart of our proposed solution lies Irys, a datachain that addresses the fundamental limitations of current AI agent architectures. Unlike traditional API-dependent systems, Irys provides a decentralized foundation for data management and storage, enabling the development of truly autonomous agents. By introducing the concept of programmable data, Irys moves beyond simple storage solutions to create an interactive layer where data can actively participate in agent operations and smart contract executions [4]. This approach not only enhances the capabilities of AI agents but also ensures cost-effective scalability - a crucial factor often overlooked in existing frameworks. Through its decentralized architecture, Irys enables agents to access, process, and interact with data in a transparent and reliable manner, effectively bridging the gap between user-friendly agent creation and autonomous capabilities. This infrastructure serves as a crucial building block for the next generation of AI agents, supporting innovations across various domains while maintaining the principles of true decentralization and autonomy.

The primary objective of this paper is to present an alternative perspective on the AI agent narrative and contribute to its evolution. By proposing a decentralized framework for multi-agent systems, we aim to demonstrate how agents can interact in a truly autonomous and decentralized manner, moving beyond the limitations of current API-centric approaches. This paper seeks to bridge the gap between user-friendly agent creation and agent capabilities, while maintaining a focus on decentralization and genuine autonomy.

## 2. System Architecture

### A. Core Components

The proposed decentralized framework for multi-agent systems is underpinned by three primary components: Providers, Orchestrators, and Workers. Each of these components plays a crucial role in ensuring seamless interaction, efficient task execution, and robust data management within a decentralized environment.

Providers are responsible for data retrieval and context augmentation. In a decentralized setting, Providers interact with the Irys datachain to access and store relevant data necessary for agent operations. By leveraging the immutable and transparent nature of blockchain-based storage, Providers ensure that the data accessed is reliable and verifiable [5]. Additionally, Providers enhance the contextual information available to agents by aggregating data from multiple sources, thereby enriching the decision-making process. This capability is vital for maintaining high levels of accuracy and relevance in agent responses.

Orchestrators serve as the nexus for request management within the multi-agent ecosystem. Their primary function is to oversee the flow of requests and responses between agents, ensuring that each request is directed to the appropriate Worker and that responses are accurately routed back to the initiating agent. Orchestrators utilize a tagging system to categorize and prioritize requests based on predefined criteria such as service category, and protocol specifications. This systematic approach not only optimizes resource allocation but also minimizes latency, thereby enhancing the overall performance and responsiveness of the system. Furthermore, Orchestrators implement validation mechanisms to verify the integrity and authenticity of the data exchanged, thereby safeguarding the system against malicious actors and erroneous data inputs.

Workers are tasked with the execution of specific operations or tasks as dictated by incoming requests. In the context of Web3, Workers can assume diverse roles such as social media influencers, DeFi operators performing token swaps, or interacting with liquidity pools to manage assets [6]. These specialized agents operate autonomously, utilizing their predefined skill sets to perform functions ranging from data analysis and processing to more complex algorithmic computations and financial transactions [7]. Upon receiving a task, a Worker engages with the Orchestrator solely to retrieve additional context from Providers. This interaction ensures that Workers have the necessary information to perform their duties effectively. After executing the required operations, Workers store the results and document each step of the process directly on the Irys datachain. This direct storage approach ensures

transparency and traceability of all actions performed, enhancing the reliability and accountability of the system.

This modular approach allows for efficient parallel processing of tasks, thereby enhancing the system's throughput and reliability. By decentralizing task execution and leveraging the immutable storage capabilities of Irys, Workers contribute to a robust and resilient multi-agent framework capable of supporting a wide range of decentralized applications within the Web3 ecosystem.

The interplay between Providers, Orchestrators, and Workers creates a robust framework that facilitates decentralized, autonomous interactions among agents. Providers ensure that high-quality data is accessible and contextually rich, Orchestrators manage the intricate web of communication and task distribution, and Workers execute tasks with precision and efficiency. Together, these components form the backbone of a multi-agent system capable of operating reliably within a decentralized, blockchain-based environment. This architecture not only addresses the limitations of current multi-agent frameworks but also paves the way for innovative applications in various domains leveraging Web3 and blockchain technologies.

## B. Communication Framework

Effective communication within a multi-agent system is paramount to ensuring seamless interactions, efficient data retrieval, and coherent task execution. The proposed framework integrates data storage and request handling within the same decentralized infrastructure, employs a tag-based system for data organization, utilizes GraphQL for querying, and delineates a clear interaction flow between components to facilitate robust and scalable agent interactions [8].

In the proposed decentralized framework, both data storage and request handling are managed within the Irys datachain. Instead of treating these processes as separate entities, the framework leverages a unified storage mechanism where both data and requests are stored as messages on the blockchain. The differentiation between data storage and request handling is achieved through a nuanced tag-based system.

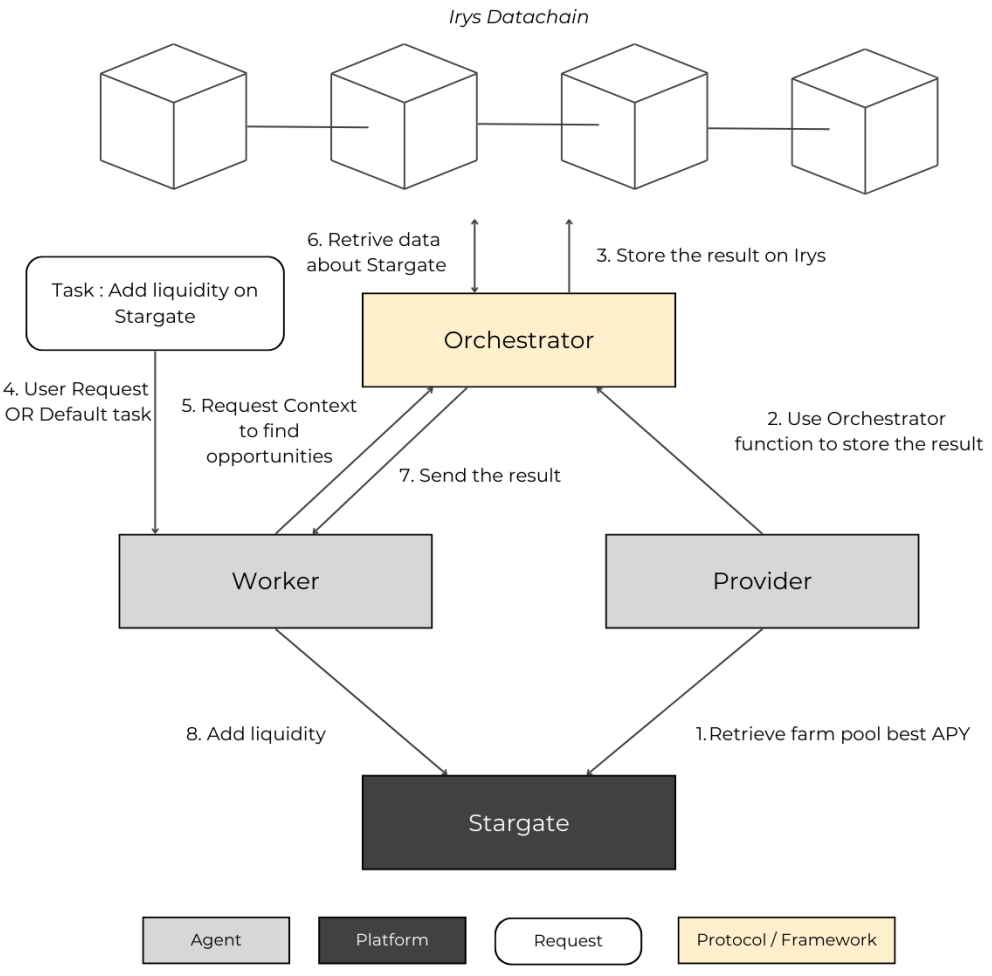
Each message stored on Irys is annotated with specific tags that categorize it either as a data message or a request message. This tagging strategy allows agents to filter and retrieve messages based on their type without the need for separate storage mechanisms. For instance, data messages might include tags such as "*MessageType: Data"* along with other relevant descriptors, while request messages would have tags like "*MessageType: Request"*. This approach ensures that both data and requests coexist harmoniously within the same storage environment, simplifying the architecture and enhancing data integrity [8].

Moreover, tags serve as metadata descriptors that categorize and index data stored on the Irys datachain. Each piece of data or request is associated with multiple tags, allowing for granular and flexible querying based on various attributes such as service category, timestamp, protocol, and reputation.

This tagging mechanism enables Providers to store data and requests in a structured and accessible manner, ensuring that relevant information can be quickly retrieved in response to agent queries. By utilizing tags, the system can filter and prioritize data based on specific criteria, enhancing the accuracy and relevance of the information provided to agents. Additionally, tags facilitate interoperability and standardization across different agents and services, promoting a cohesive and unified data environment within the decentralized network [8].

To harness the full potential of the tag-based system, the datachain integrates GraphQL as the primary query language for data retrieval. GraphQL offers a flexible and efficient means of querying data stored on Irys, allowing agents to specify precisely the information they require without over-fetching or under-fetching data [8].

By leveraging GraphQL, agents can construct complex queries that combine multiple tags and filters, enabling data retrieval operations tailored to their specific needs. This capability is particularly advantageous in a decentralized environment where data is dispersed and dynamic. GraphQL's ability to handle nested queries and its support for real-time data updates make it an ideal choice for managing the intricate data relationships inherent in multi-agent systems.



The interaction flow within the communication framework is orchestrated through a well-defined sequence of operations involving Workers, Orchestrators, and Providers. The process begins when a Worker identifies a task to explore or execute. To perform this task effectively, the Worker initiates a request to the Orchestrator to obtain additional context and relevant data from Providers.

Upon recognizing the need for more information, the Worker sends a request message to the Orchestrator by storing it on the Irys datachain with the appropriate tags (e.g., MessageType: Request). The Orchestrator continuously monitors the datachain for new request messages. Once the Orchestrator detects the Worker's request, it evaluates the request's parameters and determines the necessary context required to fulfill the task.

The Orchestrator then queries the Irys datachain using GraphQL to retrieve the relevant data from Providers based on the tags associated with the request. This data is subsequently forwarded to the Worker, enabling it to proceed with the task execution. During this process, the Worker may interact with Providers to obtain additional data or context as needed, ensuring that it has all the necessary information to perform its operations accurately.

After completing the task, the Worker stores the results and logs each step of the process directly on the Irys datachain with appropriate tags to denote them as data messages (e.g., MessageType: Data). This direct storage approach ensures transparency and traceability of all actions performed, enhancing the reliability and accountability of the system.

## 3. Security and Reliability Mechanisms

Ensuring the security and reliability of a decentralized multi-agent system is paramount to maintaining trust, integrity, and efficient operation within the network. The proposed framework incorporates comprehensive mechanisms for data quality control, focusing on orchestrator verification processes, malicious provider detection, and the implementation of a reputation system. These elements work in concert to safeguard the system against erroneous data, malicious activities, and to promote trustworthy interactions among agents.

The Orchestrator plays a critical role in maintaining data quality by overseeing the verification of data integrity and authenticity. Upon receiving data from Providers, the Orchestrator employs a multi-tiered verification process to ensure that the information is accurate and reliable. This process begins with the initial validation of data against predefined schemas and protocols using GraphQL queries, ensuring that the data conforms to the expected structure and content requirements. Subsequently, the Orchestrator cross-references the incoming data with existing records and contextual information stored on the Irys datachain. By leveraging the immutable nature of blockchain storage, the Orchestrator can verify the consistency and provenance of the data, identifying any discrepancies or anomalies that may indicate potential issues.

Detecting and mitigating the impact of malicious Providers is crucial for maintaining the overall integrity of the multi-agent system. The framework implements several strategies to identify and neutralize Providers that exhibit malicious behavior or consistently provide

erroneous data. In addition to automated monitoring, the framework incorporates community-driven mechanisms where the community can report suspected malicious behavior. Such reports are subject to verification through consensus mechanisms, wherein multiple trusted third parties must corroborate the claims before any punitive actions are taken. This decentralized approach to threat detection ensures that no single entity holds excessive power, fostering a collaborative environment for maintaining system security. Once a Provider is identified as malicious, the system enacts predefined penalties, which may include reduction of their reputation score, or permanent banning from the network in severe cases. These measures are designed to deter malicious activities and promote adherence to the system's standards and protocols.

Reputation scores influence the selection process for Providers when agents request data or task execution. Higher reputation Providers are prioritized, ensuring that the most trustworthy agents are utilized preferentially. This incentivizes Providers to maintain high standards of performance and discourages malicious or negligent behavior. Additionally, the reputation system facilitates the implementation of weighted consensus mechanisms, where the opinions and data from high-reputation Providers carry more significance in decision-making processes, thereby enhancing the overall reliability of the system [9].

Consensus and validation are critical components in maintaining the security and reliability of a decentralized multi-agent system. In a decentralized architecture, achieving consensus among multiple Providers is essential to ensure that critical tasks are executed reliably and securely. The framework employs a multi-provider consensus mechanism wherein multiple Providers are tasked with performing the same operation or supplying the same piece of data. By comparing the responses from these Providers, the system can determine the correctness and reliability of the information. For critical tasks, the framework requires a higher level of consensus, such as a majority agreement among Providers or a unanimous consensus. Providers with higher reputation scores have greater weight in the consensus process, reflecting their proven reliability and trustworthiness. This approach not only enhances the accuracy of consensus outcomes but also incentivizes Providers to maintain and improve their reputation by consistently delivering high-quality data and performing tasks reliably [10].

To further reinforce the reliability of builders (the entities that create and deploy Providers) and discourage malicious behavior, the framework implements staking and slashing mechanisms. Builders are required to stake a certain amount of tokens or other digital assets as collateral when they register their Providers on the network. This stake serves as a financial incentive for Builders to act honestly and ensure that their deployed Providers perform their duties diligently. If a Builder is found to consistently deploy Providers that submit erroneous data, fail to perform tasks, or engage in malicious activities, the system can enact slashing - the confiscation of a portion or the entirety of their staked assets. Conversely, Builders who consistently deploy reliable and high-performing Providers can receive rewards, such as additional tokens or reputation boosts. This economic model aligns the Builders' incentives with the network's security and reliability objectives, ensuring that Builders have a vested interest in maintaining high standards of behavior and deploying trustworthy Providers. [10]

In a decentralized multi-agent system, secure management of private keys is paramount to safeguarding interactions with blockchain networks and decentralized applications (dApps). The proposed framework leverages Trusted Execution Environments (TEEs) to manage private keys associated with Ethereum Virtual Machine (EVM) or Solana keypairs, ensuring that sensitive cryptographic material remains protected from unauthorized access and potential breaches. Private keys are fundamental to blockchain interactions, serving as the credentials that authorize transactions, data storage on Irys, and interactions with DeFi protocols or other dApps. The compromise of these keys can lead to unauthorized actions, data breaches, and financial losses, undermining the integrity and trustworthiness of the entire multi-agent system. Traditional key management practices often rely on software-based storage solutions, which are susceptible to attacks, malware, and human error.

Trusted Execution Environments (TEEs) provide a secure enclave within a processor that isolates sensitive computations and data from the rest of the system. By utilizing TEEs, the framework ensures that private keys are generated, stored, and used within a hardware-protected environment, significantly reducing the risk of key extraction or unauthorized access. TEEs offer several critical benefits: [11]

- Isolation: TEEs separate private key operations from the main operating system and applications, preventing malicious software from accessing or tampering with the keys.
- Integrity: TEEs ensure that the code handling private keys remains unaltered and trustworthy, maintaining the integrity of cryptographic operations.
- Confidentiality: Data within TEEs is encrypted and inaccessible to external entities, providing a high level of confidentiality for private keys.

## 4. Agent Deployment and Testing Framework

Deploying agents into a decentralized multi-agent system requires a structured approach to ensure reliability, security, and optimal performance. The proposed framework outlines a deployment and testing strategy, encompassing a dedicated testing phase and registration and monitoring mechanisms. These components work together to validate agent functionality, maintain system integrity, and foster trust within the network.

Before full-scale deployment, agents undergo a testing phase to validate their performance and integration within the network. This phase is designed to identify and rectify potential issues, ensuring that only reliable and secure agents operate within the system. Each agent is subjected to a rigorous week-long testing period, during which its functionalities, interactions, and performance are closely monitored. This period allows for the detection of bugs, vulnerabilities, and inefficiencies in a controlled environment. The Orchestrator evaluates each agent's performance based on predefined criteria, including responsiveness, accuracy, and adherence to protocols. This evaluation ensures that agents meet the network's operational standards before gaining full access. During the testing phase, users can stake tokens and participate in a voting system to endorse or reject agents. This democratic

approach empowers the community to have a say in which agents are deemed trustworthy and worthy of deployment. The framework establishes specific reliability metrics and thresholds that agents must meet or exceed during testing. Metrics such as uptime, response time, and error rates are tracked to assess each agent's reliability and readiness for deployment.

Post-testing, agents are registered and continuously monitored to ensure ongoing compliance with system standards and to maintain the integrity of the network. Each agent is represented by a dynamic Non-Fungible Token (NFT) on the Irys datachain. These NFTs serve as unique identifiers, encapsulating essential information about the agent, including its capabilities, ownership, and status. The metadata associated with each agent's NFT is regularly updated to reflect real-time performance and reputation scores. This dynamic updating facilitates accurate reputation tracking, allowing the system to assess and display each agent's trustworthiness and reliability based on their ongoing contributions and behavior [12]. A flagging system is implemented to monitor and indicate the status of each agent. Flags can denote various states such as active, under review, compromised, or deprecated. This system enables swift identification and action against agents that exhibit malicious behavior, fail to comply with network standards, or require maintenance.

This Agent Deployment and Testing Framework ensures that only high-quality, secure, and reliable agents are integrated into the decentralized multi-agent system. By combining a thorough testing phase with dynamic registration and continuous monitoring, the framework upholds the network's integrity and fosters a trustworthy environment for autonomous agent interactions.

## 5. Conclusion

The development of a secure and reliable decentralized multi-agent system is a cornerstone for advancing autonomous interactions within blockchain-based ecosystems. This framework meticulously integrates robust security mechanisms, including orchestrator verification, malicious provider detection, and a dynamic reputation system, to ensure data integrity and trustworthiness across the network. The incorporation of Trusted Execution Environments (TEEs) for private key management further fortifies the system, safeguarding critical cryptographic operations and interactions with decentralized applications (dApps).

The comprehensive Agent Deployment and Testing Framework underscores the importance of rigorous evaluation and continuous monitoring. Through a structured testing phase, which includes week-long assessments, orchestrator evaluations, and community-driven staking and voting mechanisms, the system ensures that only high-performing and trustworthy agents are deployed. The use of dynamic NFTs on the Irys datachain for agent registration and monitoring enhances transparency and accountability, while the flagging system effectively manages agent statuses to maintain network integrity.
Consensus and validation processes, reinforced by staking and slashing mechanisms for builders, establish a resilient foundation for achieving agreement and verifying operations within the decentralized environment. The introduction of random challenge implementations

adds an additional layer of security, promoting continuous compliance and early detection of malicious behaviors.

In summary, this decentralized multi-agent framework leverages the strengths of blockchain technology, TEEs, and sophisticated reputation systems to create a secure, reliable, and scalable environment for autonomous agents. By addressing critical aspects of data quality, security, and agent management, the framework not only mitigates potential risks but also fosters a trustworthy and efficient collaborative ecosystem. As the landscape of decentralized systems continues to evolve, this framework provides a robust blueprint for future advancements in multi-agent collaborations and blockchain integrations.

## References


[1] “x.com.” X (formerly Twitter). Accessed: Jan. 7, 2025. [Online]. Available: https://x.com/aixbt_agent
[2] “cookie.fun - AI Agents Index” Cookie.fun. Accessed: Jan 7, 2025. [Online]. Available: https://www.cookie.fun/
[3] “GitHub - elizaOS/eliza: Autonomous agents for everyone.” GitHub. Accessed: Jan. 7, 2025. [Online]. Available: https://github.com/elizaOS/eliza
[4] “Irys | What Irys Is.” Irys. Accessed: Jan. 7, 2025. [Online]. Available: https://docs.irys.xyz/learn/what/what-irys-is
[5] “Irys | Why Verifiability.” Irys. Accessed: Jan. 7, 2025. [Online]. Available: https://docs.irys.xyz/learn/why/why-verifiability
[6] “⚡ Actions | eliza.” Eliza.gg - Learn about the Eliza Agent Framework. Accessed: Jan. 7, 2025. [Online]. Available: https://eliza.gg/eliza/docs/core/actions/
[7]“📝 Character Files | eliza.” Eliza.gg - Learn about the Eliza Agent Framework. Accessed: Jan. 7, 2025. [Online]. Available: https://eliza.gg/eliza/docs/core/characterfile/
[8] “Irys | Querying With GraphQL.” Irys. Accessed: Jan. 7, 2025. [Online]. Available: https://docs.irys.xyz/build/d/graphql
[9] “Onchain Reputation.” Ethereum Research. Accessed: Jan. 7, 2025. [Online]. Available: https://ethresear.ch/t/onchain-reputation/16411
[10] “Proof-of-stake (PoS).” ethereum.org. Accessed: Jan. 7, 2025. [Online]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/
[11] “Trusted Execution Environments (TEE) and Compute Verifiability | Phala Network Docs.” Phala Network Docs | Phala Network Docs. Accessed: Jan. 7, 2025. [Online]. Available: https://docs.phala.network/tech-specs/multi-proof-and-verifiable-compute/trusted-execution-environments-tee-and-compute-verifiability
[12] “Irys | Dynamic NFTs.” Irys. Accessed: Jan. 7, 2025. [Online]. Available: https://docs.irys.xyz/build/d/guides/dynamic-nft