Shinkai Protocol - Draft

Robert Kornacki Nicolas Arqueros

***Abstract*—This whitepaper introduces Shinkai, a pioneering decentralized data network designed to optimize and expand the capabilities of Large Language Models (LLMs) in the AI-driven digital era. Traditional LLMs are constrained by their static training data, limiting their ability to access and process dynamically updated information. Shinkai addresses this critical gap by creating a trustless network layer that enriches the internet with AI-friendly embeddings. It incorporates a novel file system tailored for AI data management, akin to traditional hierarchical file systems but specifically optimized for efficient storage and retrieval of embeddings.**

Shinkai leverages a decentralized network of nodes, akin to platforms like Bittorrent, but with a more user-friendly approach and organized categorization of topics and websites. This structure allows for seamless and up-to-date access to a wide array of data sources, enhancing AI's ability to provide timely and contextually relevant responses. A unique aspect of Shinkai is its integration of a zero-knowledge multiparty computation protocol (MPC), ensuring the authenticity and accuracy of data and its embeddings sourced from the web.

Central to Shinkai's architecture is the Shinkai Node, pivotal for data management across the network and enabling AI agents to engage in planning with task scheduling. At its developmental stage, these agents facilitate the alignment of tasks with user needs, marking an important step in structured planning for LLMs. With the integration of PDDL handling, though still in its early stages, sets the stage for more tailored AI-user interactions, enhancing the user experience by aligning AI capabilities more closely with individual requirements.

Shinkai not only addresses the limitations of current AI tech, but also sets the stage for a more dynamic, responsive, and interconnected AI ecosystem. This whitepaper details the design, functionality, and potential applications of Shinkai, illustrating how it fundamentally transforms the AI landscape by infusing the internet with a decentralized, AI-compatible layer of intelligence and data accessibility.

***Index Terms*—Blockchain, AI, Decentralized Networks, Large Language Models, Data Embeddings, Zero-knowledge Proofs, Multiparty Computation (MPC)**

I. The Problem

As we see in the currently-fast moving AI landscape, LLMs are becoming widely available everywhere in both local use or via inexpensive cloud services. One would assume this means that all of our every-day computing tasks will instantly become drastically improved by AI, yet this simple premise ignores all of the real complexity that hides underneath. The challenge lies in the ability of these LLMs to interact with and process the vast amounts of data being constantly created. Presently, the internet caters to human users with UIs and search engine optimization for bots, but lacks the infrastructure for efficient AI operations which require vector embeddings at a minimum.

When discussing AI's future, the conversation often veers towards the centralized versus decentralized debate. However, this binary classification is insufficient because it overlooks the practicality and privacy requirements of users.

Decentralized computation has been crucial for trustless systems where transactions and interactions need to be verified without central oversight. For AI however, the interaction isn't transactional; it is instead focused on knowledge processing. This is where the centralization versus decentralization framework fails to capture the nuances of AI integration. If personal user data must be exposed to 100's of nodes run by unknown parties to achieve "decentralization" while offering no tangible benefits, it becomes quite clear decentralized computation for AI runs into a dead end.

Rather than either side of this binary being correct, Personal AI is emerging as the sought-after solution, where computation for Large Language Models (LLMs) can be executed locally on personal devices. This local execution is preferred because it avoids the increased costs and privacy risks associated with decentralized AI, which may be beneficial for blockchain applications like smart contracts and gaming, but typically does not align with the daily needs of AI users.

As Personal LLM computation becomes more affordable due to the rise of open-source models, the focus shifts to data which is not only unique but also continuously updated. Thus the true value in the AI landscape is in ongoing access to dynamic and current data, not merely the static datasets used during initial training (rather than if the LLMs are running decentralized).

Shinkai is poised to fill this gap, establishing an off-chain peer-to-peer messaging network that connects LLMs like ChatGPT with up-to-date knowledge, secure access to private information, and personal/privacy controls. Existing AI systems often fail to access current information due to reliance on top search engine results, which can be outdated (or gamed). Moreover, platforms like ChatGPT are not configured to perform personalized tasks based on highly sensitive data such as bank accounts, crypto access, emails, and medical records, due to the inherent risks and legal liabilities involved.

Shinkai, on the other hand, seeks to strike the right balance by connecting LLMs to a decentralized AI network that offers instant access to the latest data (embeddings) from the web while being fully connected to the on-chain world as well. It provides a solution that runs on users' computers, granting them full control over their digital lives without compromising their data privacy. Shinkai's approach is made possible by building a new filesystem for AI embeddings, a decentralized network of nodes providing AI data, and a zero-knowledge multi-party computation (MPC) protocol to validate data sources. This ensures that data remains confidential and under the user's control while allowing for the expansion of the AI's knowledge and skills through subscriptions to nodes within the Shinkai Network.

Furthermore, Shinkai differentiates itself by not aiming to be a better search engine but in fact an entire personal system that offers AI the foundation to perform and excel in complex tasks. This is possible via Shinkai's toolkit system, which enables AI Agents to interact with the computing stack, such as services, APIs, web search, submitting blockchain transactions, etc. By taking lessons from programming theory, Shinkai introduces typed tools with interactive plan generation, unlocking advanced multi-step capabilities—for instance, creating a report on the last six months of discussions with a partner from all Slack, Discord, and email communications.

In essence, Shinkai offers a comprehensive and private AI system that empowers users to harness AI for a wide range of real-time applications, transcending the limitations of current AI platforms. It leverages cryptographic protocols optimized for practical use, ensuring a secure, efficient, and user-centric AI experience that aligns with the needs of the modern digital landscape.

II. Solution: The Shinkai Protocol

A. *Overview*

The Shinkai Protocol is designed to be the solid foundation for LLM-based AI agents to exponentially scale their capabilities while having full access to the most up-to-date data. With a strong emphasis on user privacy, the extension of AI knowledge, and the expansion of tooling with new primitives, **Shinkai provides the needed infrastructure for AI to seamlessly fit into our everyday workflows.**

A variety of new use cases are possible on top of Shinkai:

- **Real-Time AI Knowledge Updates**: Keep AI agents up-to-date with the latest global developments across various fields such as world news, technology, health, and finance thanks to Shinkai's decentralized data AI network. This unlocks high quality responses with the most time-relevant insights and information.
- **Content Reformatting**: Transform content across formats effortlessly, from condensing long articles into summaries, converting videos into text for quick consumption, or turning text into audio for on-the-go learning.
- **Interactive Web Navigation**: Use Shinkai as a companion for web exploration, enabling users to ask clarifying questions or save web content directly into their AI's knowledge base for future use.
- **Personal AI Assistants for All Computing**: Integrate AI seamlessly & privately with all of your personal computers/phones/devices connected. Everything stays local, ensuring privacy and data security while supporting all kinds of new workflows.
- **Automated Task Management**: Schedule and automate a wide range of tasks, from email management and deadline tracking to curating custom summaries from preferred news outlets and digital content. Better yet, it can all be personalized to user preferences and delivered at just the right time.
- **Secure Information Integration**: Privately connect and manage external data sources such as emails, slack messages, Google Drive documents, health data, financial records, and more. This allows for enhanced personal AI assistance that is both secure and highly customized to individual needs.
- **Crypto Wallet Interaction**: Enhance financial management with AI that can identify optimal APY opportunities, create price drop alerts, and assist with crafting actions for crypto transactions, all while ensuring security through verification and action requests.

These use cases not only demonstrate Shinkai's capacity to move AI integration a generation forward across various domains, but also highlights the core principles of user privacy and data security. By enabling these functionalities, Shinkai sets the stage for a new era of AI tied directly into personal and professional spaces, mirroring the efficiency and automation seen in platforms like Zapier but with a focus on decentralized, AI-driven solutions.

To make these use cases possible Shinkai unlocks local execution of AI computation, advanced file parsing into data embeddings, a custom Retrieval-Augmented Generation (RAG) pipeline, among many other new innovations to provide all of the essentials for LLMs to fit users' needs. Through RAG Shinkai significantly improves AI's understanding and generation of responses by dynamically retrieving and incorporating relevant information from a vast set of documents at the time of

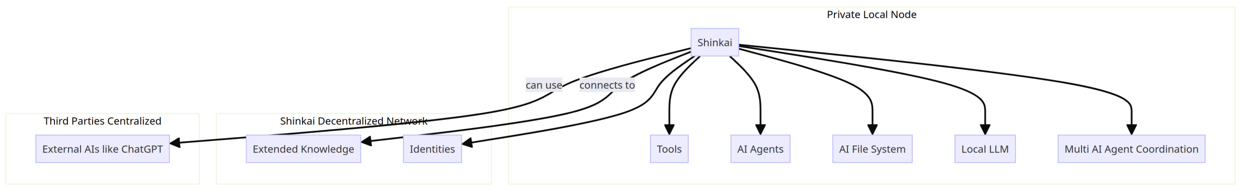

Fig. 1. Shinkai Solution

each query. This enables AI to not only rely on its initial training data but at the same time access and use the most current information available, ensuring responses are both accurate and updated.

In the digital world we find ourselves in today, timely and up-to-date information is essential for AI to actually be effective. To facilitate this Shinkai introduces a completely novel AI file system specifically designed for managing data as embeddings. This system allows for the efficient storage and retrieval of information at scale, enabling AI to broaden its knowledge beyond static data sets. Furthermore, by seamlessly integrating with Shinkai's decentralized p2p network, AI agents have immediate access to the latest data, significantly enhancing their capability to provide relevant and prompt responses.

Going past access to data and focusing on AI performing complex tasks, an additional foundational component of Shinkai's architecture is its containerized tooling system. This innovative system not only simplifies the deployment of AI use cases (by solving dependency conflicts entirely) but also incorporates a novel model of planning and execution which will encompass a wide array of tasks users perform in their every-day-lives. Furthermore by leveraging the open-source community, Shinkai consistently evolves and upgrades its capabilities via new tooling. By developing a strategic type system for tools together with interactive plan generation, Shinkai empowers AI agents to effectively manage complex tasks involving multiple steps with interactions across the computing stack, including services, APIs, code execution, interacting with blockchains, and more. This architecture markedly advances classical LLM's capabilities and scope.

B. Core Components and Functionality

The Shinkai Protocol contains several core components that underpin its innovative approach:

1) Shinkai Node: The Shinkai Node represents the personal gateway for users within the network. These nodes serve as both consumers and providers of data, contributing to a decentralized environment where collective AI intelligence is continuously enhanced through shared data and resources.

2) Decentralized Network: Central to Shinkai's infrastructure is its decentralized p2p network, which facilitates real-time sharing of AI data (ie. embeddings extracted from latest web content) and enabling AI agents to message each other. This network is supported by a bespoke file system optimized for AI data, ensuring both efficiency in data handling and scalability across the ecosystem.

3) Vector File System: At the foundation of Shinkai's data management capabilities is a vector file system specifically designed for AI embeddings. This system enables efficient and scalable storage, retrieval, and management of AI data across the entire network, facilitating seamless access to and integration of extensive data sets, while ensuring full privacy for user data.

4) Advanced RAG for Dynamic Data Access: Shinkai leverages advanced Retrieval-Augmented Generation (RAG) techniques to dynamically access and integrate the most up-to-date data with AI. This ensures that AI agents can provide responses that are not only accurate but includes online data streams which are constantly updating (ie. Social media, news, blockchains, etc.)

5) Containerized Tooling System: The introduction of a typed containerized tooling system addresses many of the common challenges deploying AI in production, such as dependency conflicts which enables streamlining the deployment of AI into every day use. This flexible typed framework supports the scheduling and execution of a broad spectrum of tasks, strengthened by the protocol's adaptability and expansion through constantly updated open-source toolkits that push the capabilities further and further.

6) Zero-Knowledge MPC Data Protocol: Through the use of zero-knowledge proofs in an Multi-Party Computation protocol, authenticity of data provided to AI can be assured without compromising privacy. This allows users to have a higher degree of trust in the responses their AI provide, first from a data sourcing point of view, which then naturally leads to a decreased rate of hallucinations with a large corpus of data available.

7) Crypto Integration: Shinkai's implementation enables AI to intelligently interact with crypto wallets & the on-chain world with a secure and robust interface. With capabilities such as hooking into multi-sig wallets, interacting with on-chain DAOs, and seamlessly integrating with up-to-date AI data, crypto use cases have a new opportunity going forward in the AI era.

8) Zero-Knowledge Computational Proofs: While the focus in Shinkai is on data, connecting verification to computation/inferencing of LLM models will eventually unlock true end-to-end assurance. The Shinkai node itself is written in Rust, and by leveraging solutions like the Succinct Processor 1 (SP1) [1] for proof generation, full fledged ZK proofs can be created for many operations a Shinkai node performs locally. It is expected that performance will continue to improve substantially.

III. Shinkai Node

The Shinkai Node is a sophisticated system written in Rust which is designed to facilitate a wide range of foundational functionalities. These include among others, AI agents, jobs, advanced plan generation execution, an AI file system, and a full fledged decentralized p2p network. As an open ecosystem, a user can simply get started by signing up via a SaaS hosting provider, or easily install the node themselves locally like any other application on their computer.

Below is a brief technical overview of the Shinkai node and its main responsibilities:

1) **Agent Management**: Orchestrates multiple AI agents for diverse tasks and roles, enabling user or agent-requested jobs.
2) **Job Handling**: Manages tasks or goals through user interactions or document processing.
3) **Networking**: Supports P2P connections with message routing, connection management, and peer discovery for efficient communication.
4) **Concurrency and Task Management**: Utilizes async tasks and threading for efficient, concurrent operations.
5) **Security and Encryption**: Ensures data integrity and unauthorized access protection through end-to-end encrypted communications and authentication.
6) **Scalability and Reliability**: Implements retry mechanisms to support message delivery reliability, and integrates merkelization among other mechanisms to enable network scaling to large numbers.
7) **Payments**: Facilitates transactions, including credit card and cryptocurrency payments, to allow AI Agents to integrate into both the Web2 and Web3 worlds.
8) **Blockchain Backed Decentralization**: Enhances security and trust with blockchain-based decentralized identities.
9) **Planning**: Features task scheduling and advanced plan generation/execution for performing every day needs of users.
10) **Vector File System**: Employs a novel technology, the Vector File System, for efficient management of data embeddings, significantly enhanced searchability of information, and underpinning the data network as a whole.
11) **Testing and Error Handling**: Incorporates a comprehensive testing suite and structured error handling for robustness.

Designed for scalability, security, and efficiency, the *Shinkai Node* lays as the foundation which unlocks the plethora of untapped AI use cases.

IV. Zero-knowledge MPC (On Demand) Data Protocol for Shinkai

On top of unlocking new use cases, one of Shinkai's goals is to enable the decentralization of AI data embeddings through providing data assurance. **The Shinkai Network will employ a zero-knowledge multi-party computation (zk-MPC) protocol to ensure authenticity of data embeddings, thereby requiring no centralized intermediaries.** This protocol will take great inspiration from the existing TLSNotary protocol [2], while adding several advancements on top specific to our AI network use case. The zk-MPC protocol is crafted to ensure the integrity and authenticity of AI embeddings' provenance, providing privacy-preserving verification which does not compromise any sensitive data (such as login credentials or financial numbers).

The verification process is designed to be flexible, allowing for both privacy and customizable levels of assurance depending on how high-value the data is deemed to be. This choice directly impacts the initial cost of creation, which will be primarily mitigated through providers setting higher delegation requirements for users on the Shinkai Network (more info about this in following sections) or optionally through direct payments/subscriptions on-chain. This ensures that Shinkai personal nodes can authenticate the data origin with confidence, upholding the network's decentralized nature and ensuring that all data used by AI is reliable.

TABLE I
Comparison of Data Embeddings Validation Options Supported By Shinkai

Protocol	**Data Assurance**	**Cost (Time + Computation)**
zk MPC On Demand + Full Validation	Very High	Very High
zk MPC On Demand + Embedding Sampling	Medium	Medium
zk MPC + Multisig Notary Proof by Multisig[1]	Medium	Medium
Just Trust Me Bro / Pinky Swear	Very Low	Ineligible

[1] The trust level can vary depending on the number and credibility of the notaries.

A. *Base (TLSNotary (zk + MPC))*

Creating proofs of data authenticity and integrity from TLS-secured websites is complex due to symmetric key encryption used after the TLS handshake. This encryption ensures secure communication by using a shared key for both encryption and decryption, but it prevents the creation of independently verifiable proofs without exposing the key, compromising data security.

In the realm of cryptographic advancements for securing data integrity and privacy, `PADO` [3] and `DECO` [4] stand out. `PADO` employs a ”garble-then-prove” technique aimed at creating on-chain proofs for data authentication over TLS, offering robust security at the cost of increased complexity. Conversely, `DECO` leverages zero-knowledge proofs to facilitate privacy-preserving data retrieval, validating data authenticity without revealing sensitive information. However, both technologies present integration challenges within `Shinkai`. The complexity and on-chain focus of `PADO` surpass the requirements for `Shinkai`’s more streamlined, off-chain proof approach. Meanwhile, `DECO`, despite its innovation, is hindered by its closed-source nature and their approach does not appear highly performant [3]. These factors contribute to `Shinkai`’s decision to utilize `TLSNotary`, enhanced with Multi-Party Computation (MPC) and cryptographic commitments, to efficiently safeguard data integrity and privacy without the limitations of `PADO` and `DECO`.

TLSNotary addresses this by introducing a third-party auditor in the handshake process, utilizing Multi-Party Computation (MPC) and cryptographic commitments. This allows the Notary to verify data’s integrity and origin without accessing the actual data or encryption keys, enabling the creation of verifiable proofs without sacrificing privacy.

For a detailed understanding of TLSNotary, including its mechanisms and integration with protocols like Shinkai, see Appendix A. This section offers an in-depth look at TLSNotary’s role in secure and private data verification. Shinkai extends this protocol to make it work with embeddings (AI data).

B. *zk MPC On Demand + Full Validation*

For users seeking complete assurance of embeddings’ provenance, Shinkai Nodes offer a full validation option. This process employs TLSNotary to validate data provenance, subsequently enabling direct validation of the embeddings created.

C. *zk MPC On Demand + Full Validation for Derivative Work*

In the future Shinkai Nodes will also wish to share embeddings of derivative works from websites, some with original content potentially locked behind sharing restrictions. Utilizing a decentralized LLM that generates a zk-proof of computation, a Node could distribute a summary of restricted content under fair use alongside a zk-proof. The zk-MPC protocol would thus validate the pieces of the original content which were used as inputs for the new derivative summary.

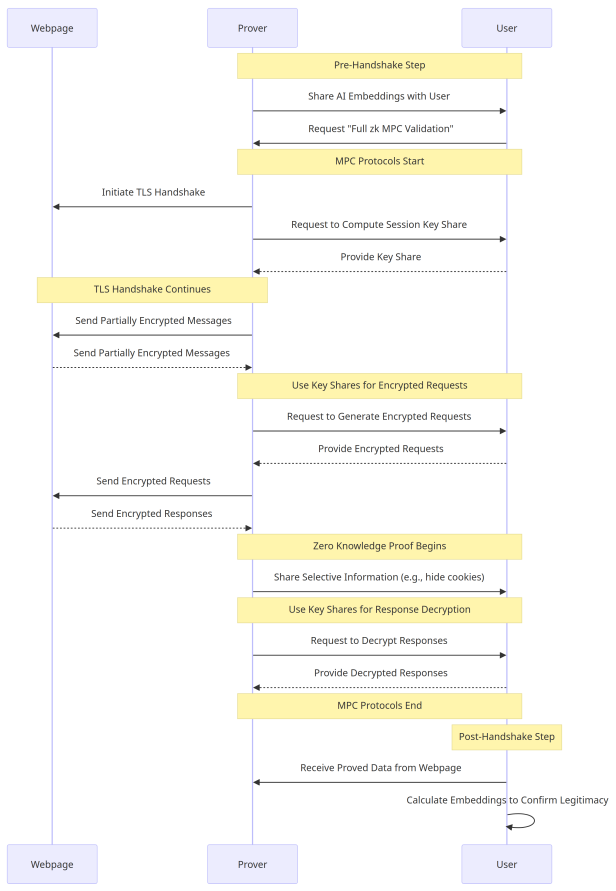
Webpage
Prover
User
Pre-Handshake Step
Share AI Embeddings with User
Request "Full zk MPC Validation"
MPC Protocols Start
Initiate TLS Handshake
Request to Compute Session Key Share
Provide Key Share
TLS Handshake Continues
Send Partially Encrypted Messages
Send Partially Encrypted Messages
Use Key Shares for Encrypted Requests
Request to Generate Encrypted Requests
Provide Encrypted Requests
Send Encrypted Requests
Send Encrypted Responses
Zero Knowledge Proof Begins
Share Selective Information (e.g., hide cookies)
Use Key Shares for Response Decryption
Request to Decrypt Responses
Provide Decrypted Responses
MPC Protocols End
Post-Handshake Step
Receive Proved Data from Webpage
Calculate Embeddings to Confirm Legitimacy
Webpage
Prover
User

Fig. 2. zk MPC Full Validation

This framework supports amalgamating content from multiple restricted sources, allowing a zk-compatible LLM to produce a comprehensive overview/analysis. This overview, paired with zk-MPC proof, affirms the source of the original content of all used sources.

D. *zk MPC + Multisig Notary Proof my Multisig*

Optimizing the zk-MPC on Demand process for reuse among multiple users begins to resemble the functionality of on-chain oracles, like Chainlink. This method employs a set of known notaries who on top of the base zk-MPC protocol, review and sign the generated data embeddings (signing their hash). The Shinkai Node providing the embeddings also shares these notary signatures with users, who can confirm the notaries’ authenticity by verifying their public signing keys on-chain through the on-chain Shinkai Registry.

E. Just Trust Me Bro / Pinky Swear

All methods have trade-offs concerning computation and verification costs, as well as the time required for production and verification. Shinkai Nodes offer AI data provenance validation services, which often may incur additional costs (either direct payment or through delegation). Some users may choose a less expensive option based on reputation, where data embeddings are offered at a lower delegation rate or even for free, promoting other paid data subscriptions for higher assurance.

V. Shinkai Vector File System

The Shinkai Vector File System, is a novel user-space file system tailored specifically for AI data management. Unlike traditional OS-level file systems, it is instead optimized for robust storage and retrieval of data directly tied to their embeddings (which are high-dimensional vector representations of the data). This is crucial for AI applications as it enhances the overall capabilities of AI while seamlessly integrating into the Shinkai Network to unlock a scalable web-of-knowledge for all AI to take advantage of (fully permissioned & privacy-respecting).

A. Advantages of A Vector File System Over A Vector Database

Unlike existing Vector Databases (which current generation AI projects use), the VectorFS is designed to fit into the modern computing/web stack naturally, allowing deep composability, resusability, and thus converting much of the existing internet into data embeddings. The advantages of the VectorFS include:

- **Hierarchical Organization**: The `VectorFS` employs a hierarchical structure that mirrors real-world data relationships, unlike flat data models in databases.
- **Granular Access Control**: Provides detailed access controls at various levels, enhancing security and allowing teams/companies to work internally & collaborate externally as well.
- **Data Integrity and Security**: Utilizes Merkle trees for secure integrity checks, facilitating quick verification and tamper detection.
- **Optimized For Quality**: Features specialized indexing and search architecture optimized for returning quality vector search results.
- **Domain-Specific Customization**: Can be tailored with domain-specific optimizations or using multiple customized embedding models, enhancing search quality.
- **Optional Auditability and Compliance Support**: Supports comprehensive access logs, allowing optional support for use cases with high compliance and data governance standards.
- **Distributed Computing Compatibility**: Built for distributed environments, unlocking data consistency across multiple nodes or clusters.

B. Vector File System Path Model

At the heart of the Vector File System is a hierarchical structure based on paths, where root '/' represents the root directory. This structure allows for an intuitive organization of data, mirroring the familiar file system hierarchy found in operating systems.

Every non-root path contains a file system entry, which can be either a folder or an item:

- **Folders**: These are directory-like structures that can exist at any depth within the VectorFS, starting from the root. A folder can contain other folders (subdirectories) and items (files), allowing for a nested, tree-like organization of data. Folders in VectorFS are not just simple containers; they also hold metadata such as creation, modification, and access times, providing a rich context for the data they contain.
- **Items**: Representing the "files" within the VectorFS, items are containers for a single data embedding file (Vector Resource) + an optional set of source documents the embeddings were created from (pdfs, docs, txts, etc). Items enable the VectorFS to tie original file formats with their data embedding representations, enhancing the system's utility for a wide range of applications.

C. Data Storage in VectorFS

Items in the `VectorFS` store a broad range of information to enable both high quality vector search results, and seamless integration into countless use cases:

- **Top-level Vector Embedding**: Stores high-dimensional vector embedding that acts as a summary for the whole Vector Resource/internal data embeddings inside. Improves VectorFS-wise searches.
- **Internal Node Vector Embeddings**: Vector Embeddings which map onto the pieces of actual content, which are used by the AI Agents themselves.
- **Metadata Index**: Stores common info such as page numbers or other use case specific metadata internally, while having a top-level index to allow for extremely efficient metadata searching across the VectorFS.
- **Permissions**: Contains advanced read/write permissions with whitelisting support which connects directly into Shinkai's decentralized p2p network. Unlocks pay-to-access content via crypto as well.
- **Timestamps**: Holds various timestamps that provide a high degree of transparency whenever items are updated, mutated, moved, or read. Useful for

syncing data across the network, and connecting to external applications.
- **Merkle Hashes**: Utilizes Merkle hashes for data integrity verification.

D. *VectorFS As A Merkle Tree*

Merkle hashes serve to cryptographically fingerprint large data structures, allowing swift integrity verification. They are often used in blockchain contexts as a great primitive that underpin the trustworthiness of data.

Thanks to it's hierarchical nature the VectorFS implements full merkelization, where every folder and item stored has its own merkle root. This acts as an extremely pivotal benefit in the world of decentralized AI data. **All p2p syncing and data exchange protocols can take advantage of the fact the entire VectorFS itself can be represented as a Merkle tree, allowing for numerous scaling and distribution optimizations to be made.**

Unlike classical VectorDBs which cannot support this functionality, the VectorFS is built from the ground up to shine in the decentralized AI Era.

VI. Decentralized Shinkai Identities and Their Smart Contract

The Shinkai Network integrates directly into blockchain to implement a decentralized and censorship-resistant system for Shinkai identities. Storing public node data on-chain ensures that Shinkai nodes within the ecosystem can effortlessly discover and communicate with each other. The decentralized Identity Registry smart contracts are central to Shinkai, enabling secure and autonomous interactions for managing identities with no middleman required.

A. *Shinkai Identities*

As mentioned above, at the core of the Shinkai Network is the concept of Shinkai Identities. These identities serve as unique "names" for users/nodes on the network, akin to a domain name, but with broader functionalities. By staking Shinkai tokens, with the amount required inversely proportional to the length of the identity name, users can register an identity on-chain (ex. @@alice.shinkai). This inversely proportional relationship ensures that shorter (desirable) names are harder to acquire, thereby preventing name squatting while encouraging the efficient use of the whole namespace.

B. *Key Features of Shinkai Identities*

- **Decentralized Identity Registration:** Identities (as NFTs) are registered/created by staking Shinkai tokens, with the amount required inversely proportional to the name's length, prioritizing the efficient use of the namespace.
- **Ownership and Transferability:** Registered identities are owned and held in a user's wallet as an NFT, with the capability to transfer ownership of the identity (and thus associated stake) to other addresses.
- **Connects With The Shinkai Node:** Your identity connects directly with your Shinkai node, allowing all messages, files, or anything else anyone (or any AI Agent) sends to you to seamlessly arrive at your node.
- **Dynamic Management:** Owners can adjust their stake, withdraw excess tokens without forfeiting the identity, and fully unstake to relinquish ownership (thereby burning the NFT).
- **Infinite Sub-identities:** Underneath a single Shinkai identity, an infinite number of sub-identities can be created for all devices/computers/AI agents that the user owns.
- **Router/Proxy Nodes:** By specifying one or more router/proxy nodes the network supports further scalability, enhanced message delivery assurance, and privacy.
- **Batch Operations:** Facilitates the management of multiple identities through batch registration, transfer, and unstaking processes.

C. *Sub-identities For Integration With User Devices*

Sub-identities enhance the Shinkai Network's flexibility, allowing intricate management of devices, agents, and various functionalities within the Node. In fact they even allow multiple distinct people (ie. friends/family) to have their own sub-identities which map onto different profiles in the Shinkai node.

This design supports advanced access controls while enabling AI Agents to have their own sub-identities as well (unlocking the ability to p2p message each other). Here are examples of sub-identities on Shinkai:

- `@@Alice.shinkai/profileName` – a user profile within the network.
- `@@Alice.shinkai/profileName/agent/myChatGPTAgent` – an AI agent, linked to the user's profile.
- `@@Alice.shinkai/profileName/device/myPhone` – a specific device like a smartphone, associated with the user's profile.

The Shinkai naming convention follows a standard of: `@@baseIdentity/profileName/subidentityType/subidentityName`, where:

- `baseIdentity` is the user's Shinkai identity assigned on-chain in the Registry.
- `profileName` specifies a user-created profile.
- `subidentityType` categorizes the sub-identity as either an agent or a device.

- `subidentityName` provides a unique identifier for the sub-identity.

This sub-identity model allows a Shinkai node to scale across countless use cases. **Whether a user will have 50 devices connected in an IoT cloud, 150 AI agents running in parallel communicating with each other/other agents, or 12 friends/family members each with their own profile, Shinkai is designed to support all of such use cases seamlessly.**

D. Routing Nodes and Network Resilience

Routing nodes offer a strategic solution to the challenges often faced with direct P2P connections. These nodes allow users looking for enhanced privacy (hiding the public IP of their node) or ease of use via tunneling through restrictive firewalls while also ensuring consistent message delivery across the network (backed by end-to-end encryption).

Furthermore, nodes with extremely high demand may utilize multiple routing-nodes as balance loaders to ensure high connectivity even under serious traffic. All together, routing nodes offer an on-network native solution to overcome classical P2P issues.

E. Shinkai's On-Chain Identity Registry and Delegation Mechanism

Shinkai's on-chain Identity Registry supports all core capabilities for managing and operating Shinkai Identities in a fully decentralized manner. Each identity lives on-chain and contains information about:

- **Staked Tokens:** Reflects the Shinkai tokens staked underneath the identity. Each identity's staked tokens are distinct, ensuring isolated management and facilitating specific actions like partial withdrawals or stake increases without affecting other identities.
- **Node Addresses/Proxy Nodes:** Critical for the network's decentralized infrastructure, allowing identities to specify direct node addresses or utilize proxy nodes for enhanced connectivity and privacy.
- **Encryption and Signature Keys:** Underpinning secure communications across the network, these keys ensure nodes can decrypt and validate who they are sending/receiving messages from.
- **Tokens Delegated:** The amount of Shinkai tokens the identity has delegated to others.
- **Incoming Delegations:** The amount of Shinkai tokens other identities on the network are delegating to the identity.
- **Records HashMap:** Provides a flexible datastore for adding additional information related to an identity, enabling data attestation and extensibility for other protocols.

The delegation of tokens is a cornerstone of Shinkai's economy, enabling identities to support other nodes/service providers on the network with no added personal cost.

F. Shinkai's On-Chain Architecture

As mentioned in the previous section, Shinkai's on-chain architecture underpins the ecosystem, enabling a range of operations from identity registration to stake management and token delegation. The system is designed with upgradability in mind, ensuring the network can adapt and evolve over time. Key components of the on-chain architecture:

- **ShinkaiRegistry:** The central hub which allows for managing identities, staking, delegation, and node information.
- **ShinkaiToken:** Facilitates staking and delegation, enabling Shinkai's identity namespace to function.
- **ShinkaiNFT:** Represents registered digital identities as unique NFTs, providing a tangible asset for ownership and transferability within the ecosystem.

This on-chain architecture provides a robust framework for decentralized identity management, empowering users with control, security, and flexibility. Additionally, it is not merely limited to Ethereum but will extend to other ecosystems and Layer 2 solutions ensuring future-readiness no matter what chains end up dominating the crypto landscape going forward.

VII. Networking and The Shinkai Message

For many projects networking and messaging details may just be a footnote. However for Shinkai, with multiple systems interacting with the Node, it's crucial to explain the core mechanics at play that the Node interacts with and supports.

- **Node-to-Node Communication**: Utilizes TCP for the transmission of `ShinkaiMessage` as the core primitive. Also supports sharing an AES-GCM symmetric key as a session key for direct file transfers.
- **Node API Communication**: Implements REST APIs that leverage `ShinkaiMessage`, with strong security through encryption and message signing which guarantees Shinkai identity verifiability. Nodes and clients must synchronize their clocks to prevent the rejection of old messages. To avoid message replay, the Node records the hash of each message, ensuring they cannot be reused. Additionally, clients can share an AES-GCM symmetric key within an encrypted `ShinkaiMessage` for easier file uploads. WebSocket (WS) connections, secured with AES-GCM, facilitate real-time updates between Nodes and Clients.

A. *ShinkaiMessage Structure*

A `ShinkaiMessage` is a complex structure designed to facilitate secure and versatile communication and data transfer within the Shinkai ecosystem. It is comprised of several components, each tailored to specific aspects of message handling, encryption, and metadata management.

1) Body (`MessageBody`): The `MessageBody` carries the actual content of the message, supporting encrypted and unencrypted data:

- **Encrypted (`EncryptedShinkaiBody`)**: Uses ChaCha20Poly1305 for data encryption, ensuring secure transmission. This is the global default used for Shinkai Messages ensuring privacy for all communications.
- **Unencrypted (`ShinkaiBody`)**: Directly includes the message data (`MessageData`) and internal metadata (`InternalMetadata`). This is only suitable for specific use cases that require extremely high throughput of non-sensitive information. This is an optional setting available, not used by the network by default.

2) External Metadata (`ExternalMetadata`): Encapsulates essential delivery and verification information:

- **Sender and Recipient**: Shinkai identities of the message originator and target recipient.
- **Scheduled Time**: Timestamp for when the message is to be sent/processed.
- **Signatures**: Utilizes the Ed25519 algorithm for signing, thereby verifying the message's authenticity and integrity.
- **Additional Fields**: Allows for custom use cases or internal processing needs.

3) Encryption (`EncryptionMethod`): Details the encryption used for the message. Currently supports (x25519_dalek) for key exchange and ChaCha20Poly1305 for encrypting message content.

4) Version (`ShinkaiVersion`): Indicates the `ShinkaiMessage` format version, ensuring compatibility and facilitating a clean upgrade-path for the future.

Note: The keys used for encryption and signing within the `ShinkaiMessage` are distinct, enhancing security by keeping the encryption of message content fully separate from the verification of message authenticity.

Each component of the `ShinkaiMessage` plays a crucial role in securely transferring data within the Shinkai ecosystem by leveraging state-of-the-art cryptography for both signing and encryption.

B. *Lifecycle and Example of ShinkaiMessage Communication*

Let's take a look at the process of sending a message between two users, Alice and Bob, on the Shinkai network. Alice wishes to send the message "Welcome to Shinkai!" to Bob. This involves several steps & interactions including with Alice's device, Alice's Shinkai Node, the Blockchain, Bob's device, and Bob's Shinkai Node.

1) **Identity and Key Discovery**: Alice retrieves Bob's Node information (IP, port, or router/proxy) and public encryption keys from the Identity Registry on the blockchain (using Bob's identity @@bob.shinkai), and queries Bob's Node for his device's keys as well.
2) **Message Preparation**: Alice crafts a ShinkaiMessage with an inner layer targeting Bob's device and an outer layer for Alice's Node. The inner layer can be decrypted with Bob's device encryption key, while the outer layer via Alice's Node encryption key, following a Diffie-Hellman key exchange.
3) **Signing and Sending**: Alice's device signs both the inner and outer layers of the message to confirm its authenticity and sends it to her Shinkai Node.
4) **Node Processing (Alice's Side)**: Alice's Node decrypts the outer layer and verifies Alice's signature. It then uses Bob's Node information from the blockchain to re-encrypt the outer part of the message, but now targeting Bob's Node. This re-encryption ensures a high level of privacy and limited data leakage.
5) **Transmission to Bob's Node**: The message is sent over the network from Alice's Shinkai Node to Bob's Shinkai Node.
6) **Node Processing (Bob's Side)**: Bob's Node decrypts the outer message, verifies the signature against Alice's Node information on the blockchain, and securely stores the message for Bob to read from one of his devices.
7) **Delivery to Bob**: When Bob connects to his Node, the message is delivered to his device, and the inner message is finally fully decrypted and read. This provides end-to-end encryption, and allows for private communication to be possible between users even when using 3rd party hosting providers for the Shinkai Node.

This process is visually depicted in Fig. 1, "Overview of Networking Flow on Shinkai", illustrating the secure and efficient messaging that Shinkai nodes perform.

VIII. Planning, Reasoning, and Toolkits

In the realm of artificial intelligence, the process of making plans and logical deductions using Large

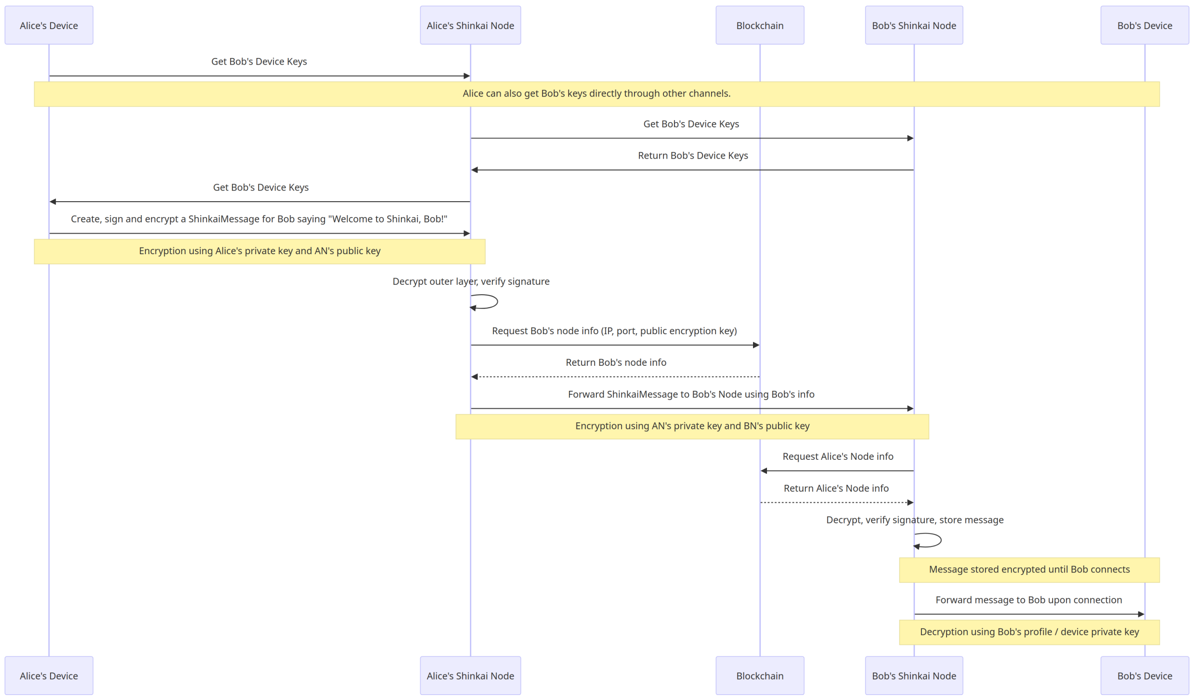
Alice's Device
Alice's Shinkai Node
Blockchain
Bob's Shinkai Node
Bob's Device
Get Bob's Device Keys
Alice can also get Bob's keys directly through other channels.
Get Bob's Device Keys
Return Bob's Device Keys
Get Bob's Device Keys
Create, sign and encrypt a ShinkaiMessage for Bob saying "Welcome to Shinkai, Bob!"
Encryption using Alice's private key and AN's public key
Decrypt outer layer, verify signature
Request Bob's node info (IP, port, public encryption key)
Return Bob's node info
Forward ShinkaiMessage to Bob's Node using Bob's info
Encryption using AN's private key and BN's public key
Request Alice's Node info
Return Alice's Node info
Decrypt, verify signature, store message
Message stored encrypted until Bob connects
Forward message to Bob upon connection
Decryption using Bob's profile / device private key
Alice's Device
Alice's Shinkai Node
Blockchain
Bob's Shinkai Node
Bob's Device

Fig. 3. Overview of Networking Flow on Shinkai

Language Models (LLMs) has revealed a large swath of complexity which LLMs cannot solve by themselves today. This complexity arises from the vast computational and architectural challenges intrinsic to these tasks. As the number of variables in a given problem grows, so does the difficulty of creating and verifying a plan, leading to a classification known as PSPACE-complete. The unpredictability of task execution in real-world scenarios further adds to this complexity, often resulting in issues akin to those encountered in stochastic planning where outcomes are uncertain.

This pivotal dilemma in AI agent automation, highlighted in the interview with Subbarao Kambhampati titled *”Planning, Reasoning, and Interpretability in the Age of LLMs”* [5], revolves around whether to limit focus to problems with tractable computational complexity or to embrace the full spectrum of problems, striving for the best possible solutions without guarantees. This has historically divided the AI community into ”neat AI”, focusing on tractable logic to avoid prohibitive inference costs, and ”scruffy AI”, which accepts the challenges of complexity and intractability.

The limitations of Large Language Models (LLMs) become apparent in planning and reasoning due to their reliance on approximate retrieval & responses rather than genuine reasoning, thereby necessitating external verification to ensure plan correctness. This underscores the significance of a more nuanced approach towards AI planning, advocating for a balance between embracing complexity and striving for computational efficiency.

A. Path Forward

To address these challenges, a promising solution is available involving a hybrid approach that utilizes Large Language Models (LLMs) for generating initial ideas, complemented by external validation to refine and confirm the accuracy of these plans. This method employs a process known as iterative refinement or ”back prompting” together with typed tools, which together systematically improve the quality and practicability of plans created by LLMs. In this process, LLMs generate preliminary plans that first pass through type validation, and then are reviewed by external verifiers. Based on feedback, LLMs receive new prompts that incorporate the needed changes specified, enabling them to adjust and improve their initial plans. This cycle of generation, feedback, and refinement continues until a satisfactory plan is achieved, ensuring that the final output is both feasible and of high quality.

This path forward, as initially touched upon in the paper *”LLMs Can’t Plan, But Can Help Planning in LLM-Modulo Frameworks”* [6], focuses on a pragmatic use of LLMs to address the AI planning problem. By emphasizing a collaborative synergy between machine static analysis, external verifiers, and the innovative capacity of LLMs, we are able to create a robust, adaptable, and efficient framework for tackling complex planning and reasoning tasks in AI.

IX. Shinkai Toolkit System

A. Overview

To overcome the previous listed challenges in AI planning and execution, Shinkai offers a streamlined framework for the creation, management, and deployment of specialized toolkits. These toolkits, designed for LLM Agents, enable a broad range of functionalities, including code execution, data fetching, API calling, and file management, thereby expanding the capabilities of Large Language Models (LLMs) beyond their inherent limitations.

At the core of the toolkit system are the following set of work streams which are moving forward in parallel:

1) **Containerized Tooling**: Shinkai implements tool encapsulation to eliminate dependency conflicts and enhance security, allowing AI agents to efficiently perform a broader array of tasks. This modular and isolated environment ensures tools are portable and adaptable, unlocking rapid ecosystem development and deployment for AI-driven applications.
2) **Synthetic Data Generation**: A new workflow is being built to generate high quality QA of external tooling usage for LLMs. Unlike existing solutions that suffer from the normal distribution problem (too much of certain data and too little of others), this new workflow addresses the problem by using a randomness sampling technique. Furthermore, the QAs are also validated using verifiers in order to avoid fine-tuning models with low-quality examples that are lacking. This stream will primarily focus on crypto tooling like TheGraph to address the major need of current LLMs lacking training to properly interact and plan for the crypto world.
3) **Types For Tools**: In order to use static analysis to ensure AI generated plans are valid and can be completed, tools are typed. This approach takes lessons from the world of programming language theory to off-load as much validation as possible from both the LLMs and humans involved, instead relying on mechanistic verification where possible.
4) **Plan Validation & Execution Runtime**: LLMs by themselves cannot validate or execute plans without a runtime that guides them while performing all plan static analysis, input validation, retry pathing, and tool execution automatically. This runtime exists within the Shinkai node and be the foundation which allows advanced tasks to be accomplished.

At the heart of the system are two variants of tools: Embedded Tools, which expose internal/core functionalities of the Shinkai node to the AI agent, and External Tools, which execute code and can perform any task such as interface with external APIs.

The Embedded Tools are part of the "Shinkai Core Toolkit," which is included automatically within every single Node. This toolkit equips AI Agents with essential capabilities on the Shinkai Network such as messaging, file/content vector search, storage,scheduling future execution, tool vector search, and more.

External Tools on the other hand are designed to be modular and extensible, allowing your AI Agents to continue growing more capable as new toolkits are installed. Anyone can create their own external toolkits enabling an ecosystem to develop with countless of use cases covered for a variety of domains:

- **Web Access**: Connect to the existing web, allowing your AI Agent to use services, web apps, and everything in between.
- **Crypto Tooling & Infrastructure**: Provide secure crypto capabilities like issuing transactions, voting on DAOs, performing trades, comparing APY opportunities, and more.
- **Financial & Accounting Software**: Deal with the world of traditional finance and hooking into/streamlining accounting processes.
- **Marketing Tools**: Offer tools for market analysis, campaign management, customer engagement, and more.
- **File Storage & Management**: Connect to existing storage providers like Google Drive, allowing your AI to manage them and unifying all your data together with the VectorFS.
- **CRM (Customer Relationship Management)**: Streamline existing CRM processes with directly hooking in AI Agents with customer docs stored in the VectorFS available to be used.
- **Calendar & Scheduling**: Simplify scheduling and event management across your different apps/services.
- **Computation**: Allow your AI Agents to gain new capabilities like executing code (isolated), performing mathematical calculations, validating cryptographic signatures/zk proofs, and more.
- **Custom Integrations**: Enable seamless integration with any external services and APIs.

With these toolkits users will be able to upgrade the capabilities of their AI as easily as installing an app on their phone. Anyone can easily code new toolkits in mere hours, unlocking novel capabilities for AI Agents.

To enable toolkits to work seamlessly with users' Shinkai nodes and enable developers to have a streamlined experience getting started, the following components are being built in the Shinkai ecosystem:

B. *Toolkit-Builder*

The toolkit-builder is a scaffolding tool that kickstarts the development of new toolkits by generating a base project structure. This structure includes a sample tool and configuration files necessary for building, testing, and deploying the toolkit. The toolkit-builder simplifies the initial setup process, allowing developers to quickly move to the implementation phase.

C. *Toolkit-Executor*

The toolkit-executor is a versatile component that facilitates the execution of compiled toolkits. It supports two modes of operation: a standalone executable mode for direct command-line execution and a web server mode that listens for execution requests over the network. This dual-mode operation enables both local and remote execution of toolkit functions, providing flexibility in how toolkits are deployed and utilized.

D. *Toolkit-Lib*

Toolkit-lib serves as the foundational library for toolkit development, offering core functionalities and interfaces required by external tools. It includes utilities for introspection, input validation, and other common tasks, thereby standardizing the way toolkits are developed and ensuring consistency across the ecosystem.

E. *Toolkit-Registry*

The toolkit-registry maintains a comprehensive record of all available toolkits within the ecosystem, including their metadata, versions, and tool descriptions. It acts as a centralized repository that facilitates the discovery and integration of toolkits into projects. The registry is automatically updated to reflect the latest changes and additions to the ecosystem, ensuring that all users have access to upgrade their AI agents' capabilities.

X. SHINKAI COMPATIBILITY

A. *Node API Design*

The goal of Shinkai is to allow every user to securely and easily connect and use their AI from anywhere. Whether they want to plug AI into their web browser, mobile phones, desktop apps, or 3rd party services, all of these use cases need to work seamlessly.

To support all of these the Shinkai team has developed multiple libraries in Rust, WASM, Python (utilizing pyo3), and native Typescript. This diverse set of targets enables the creation of applications and integrations across a wide range of platforms including iOS, Android, Chrome Extensions, desktop applications and more. Additionally, Shinkai has been integrated into various services such as Zapier, Replit, Slack, Discord, among others. This cross-platform approach ensures that Shinkai can maintain a broad and adaptable presence across different ecosystems, and allow the open source developer community to naturally flourish.

B. *Enterprise*

Enterprises are known for having tailored needs regarding specific security, operational, and regulatory requirements. Most centralized AI solutions require full disclosure of all data, running into clear conflicts for how AI should naturally fit into these workflows.

Shinkai's architecture leverages a strong privacy-oriented technological foundation that naturally aligns with the stringent requirements of enterprise environments (despite not targeting this use case in the first place). Its low-level implementation of key tech allows for more customizability and thus fitting into direct paths for enterprise to take advantage of.

For example, Shinkai's device and identity management capabilities enables fully compartmentalized file and permissions access that securely scales no matter how many users are required for a given organization. Furthermore, this neatly fits into cases where cross-organization cooperation is required as well, with fine-tuned access requirements that keeps everything in line.

Flexibility in deployment is another inherent quality of Shinkai's technological stack. The platform's architecture supports on-site deployments, enabling enterprises to integrate their own security and identity management systems on top. This adaptability ensures that Shinkai can be molded to meet the exact standards and bespoke needs of any enterprise environment.

In essence, Shinkai allows enterprises to take advantage of the latest and greatest innovations in AI, while molding into their existing technological and regulatory stack. With Shinkai's unique blend of security, scalability, and interoperability, it offers enterprises a potent tool for entering into the AI Era.

XI. LONG TERM DEVELOPMENTS

The Shinkai Network lays down the foundation for a fruitful open-source ecosystem to grow. With strong fundamentals, there are multiple paths of research and development that are possible for the ecosystem to take to continue growing going forward:

1) Embedding LLMs into the Node directly, once the state-of-art Open Source models reach a sufficient intelligence threshold to not require constant weekly/monthly updates to keep up
2) Improvements to the zk MPC protocol
3) Updates to better Embedding models with good trade-offs in speed to generate vs. storage size vs. quality.
4) Implementation of new capabilities leveraging zk for services provided by Shinkai nodes

XII. Conclusion

In conclusion, the Shinkai Protocol melds cryptographic & networking innovations together within a single system to address and extend beyond the current limitations of AI. By establishing a decentralized data network that infuses the internet with AI-friendly embeddings, Shinkai enables LLMs to access and process dynamic, up-to-date information. This is further bolstered by the use of zero-knowledge MPC protocols to provide data assurance while drastically expanding LLMs' knowledge in a privacy-preserving manner, setting the stage for a new paradigm of secure and decentralized AI.

References

[1] "SP1: Performant, Open-Source Zero-Knowledge Virtual Machine," https://github.com/succinctlabs/sp1, 2024, documentation for SP1 users and developers. SP1 is a performant, 100% open-source, contributor-friendly zero-knowledge virtual machine (zkVM) that verifies the execution of arbitrary Rust (or any LLVM-compiled language) programs. License: MIT / Apache 2.0.

[2] TLSNotary, "TLSNotary: Proof of data authenticity," Public good and open source project supported by the Ethereum Foundation under the Privacy and Scaling Exploration Team, n.d., available: https://tlsnotary.org.

[3] X. Xie, X. Wang, K. Yang, and Y. Yu, "Lightweight authentication of web data via garble-then-prove," *Preprint*, 2023.

[4] F. Zhang, H. Malavai, S. Goldfeder, and A. Juels, "Deco: Liberating web data using decentralized oracles for tls," *Cornell Tech*, 2023, available: https://arxiv.org/abs/2009.03875.

[5] S. Kambhampati and D. Bashir, "Planning, reasoning, and interpretability in the age of llm," The Gradient Podcast, 2024, episode 110. [Online; accessed 20-February-2024]. [Online]. Available: https://www.youtube.com/watch?v=nwQKuY9_Ky4

[6] S. Kambhampati, K. Valmeekam, L. Guan, K. Stechly, M. Verma, S. Bhambri, L. Saldyt, and A. Murthy, "LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks," 2024, arXiv:2402.01817 [cs.AI].

[7] K. Valmeekam, S. Sreedharan, M. Marquez, and S. Kambhampati, "On the Planning Abilities of Large Language Models : A Critical Investigation," November 2023, school of Computing & AI, Arizona State University; Department of Computer Science, Colorado State University.

[8] M. Balduccini, D. Magazzeni, and M. Maratea, "PDDL+ Planning via Constraint Answer Set Programming," Presented at the 9th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2016), New York City, USA, 16 October 2016, 2016, arXiv:1609.00030 [cs.AI].

[9] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large Language Model Connected with Massive APIs," 2023, arXiv:2305.15334 [cs.CL].

[10] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," 2022, arXiv:2201.11903 [cs.CL].

[11] P. von Platen, "How to generate text: using different decoding methods for language generation with Transformers," Hugging Face Blog, March 2020, edited on July 2023 with up-to-date references and examples. [Online]. Available: https://huggingface.co/blog/how-to-generate

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017, arXiv:1706.03762 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1706.03762

[13] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976, invited Paper.

[14] D. J. Bernstein, "Chacha, a variant of salsa20," 2008, available at https://cr.yp.to/chacha.html.

[15] ——, "The poly1305-aes message-authentication code," *Fast Software Encryption*, vol. 3557, pp. 32–49, 2005.

[16] Y. Nir and A. Langley, "Chacha20 and poly1305 for ietf protocols," RFC Editor, RFC 7539, May 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7539.txt

[17] A. Langley, M. Hamburg, and S. Turner, "Chacha20 and poly1305 for ietf protocols," RFC Editor, RFC 8439, June 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8439.txt

Appendix A
Appendix: Detailed Overview of TLSNotary Protocol

The TLSNotary protocol introduces a third-party notary into the TLS handshake process to verify the integrity and origin of data transmitted over a TLS connection without compromising privacy. This appendix provides a comprehensive overview of the protocol's operational workflow, cryptographic foundations, and integration with the Shinkai AI Network.

A. Operational Workflow of TLSNotary

1) Initiation: MPC-TLS Phase: The protocol starts with an MPC phase, where the Prover and Verifier collaboratively initiate a secure connection to a TLS-enabled server. This setup ensures that the Verifier observes only encrypted application data, maintaining the confidentiality of the exchange.

2) TLS Handshake: The TLS handshake involves cryptographic operations performed via MPC, resulting in the computation of session keys. These keys are generated in a manner that ensures neither the Prover nor the Verifier has full knowledge, preserving privacy and security.

3) Data Encryption and Decryption: Data sent to the server is encrypted using the session key, derived from both the Prover and Verifier's inputs, while incoming data is decrypted in a similar collaborative manner. This ensures the confidentiality of the transmitted data and the integrity of received data.

4) Notarization Process and Verification of Data Provenance: The Prover generates authenticated commitments to the plaintext data, which are signed by a Notary. This process facilitates the later verification of the data's origin and integrity by an application-specific Verifier, ensuring non-repudiation.

B. Cryptographic Foundations

1) Mathematical Formulations: The protocol's security is underpinned by complex mathematical operations:

- **GHASH Output Computation**: Essential for MAC computation, defined as

$$\begin{aligned} GHASH(H, & X_1, X_2, \ldots, X_m) = \\ & X_1 \cdot H^m \oplus \\ & X_2 \cdot H^{m-1} \oplus \\ & \ldots \\ & X_m \cdot H, \end{aligned}$$

  where $\cdot$ and $\oplus$ represent multiplication and the XOR operation in the finite field $GF(2^{128})$, respectively.
- **Secure Two-party Computation for MAC**: Ensures that neither party learns the other's key share, crucial for maintaining the confidentiality of the MAC.

2) Ciphertext Block Computation and Commitment Scheme: The protocol's approach to ciphertext block computation and its commitment scheme ensures data integrity and privacy. These include the conversion of additive shares into multiplicative shares and the utilization of a Merkle tree structure for commitments, enhancing the security of the data exchange.

C. Integration with Shinkai AI Network and Ensuring Trustlessness

The integration of the TLSNotary protocol with the Shinkai AI Network significantly enhances the network's capability to maintain the assurance of data embeddings. This is pivotal in a decentralized setting where verifying the authenticity and integrity of data, without compromising on user privacy, is crucial. By utilizing cryptographic constructs such as garbled circuits and oblivious transfer, the TLSNotary protocol provides a robust framework for secure computations and data exchanges across the network.

This integration is instrumental in guaranteeing that data shared through the Shinkai AI Network can be trusted without requiring centralized validation. The data assurance afforded by the TLSNotary protocol means that participants in the network can rely on the integrity and authenticity of data, facilitating a truly decentralized ecosystem for AI-driven applications and services. The protocol ensures that each piece of data exchanged within the network is accompanied by verifiable proofs, thus upholding the core principle of decentralization — eliminating the need for trust in a central authority.

D. Conclusion

The TLSNotary protocol incorporates advanced security measures, including the DEAP protocol and finite-field arithmetic to safeguard against malicious threats and ensure the robustness of MPC operations.

With the above stated, it becomes quite clear that through TLSNotary's protocol design and cryptographic underpinnings, its role within the Shinkai Network provides a strong technical foundation for secure data embedding sharing.