

LarkAI v1.0 Whitepaper

Integrated AI Network

LarkAI v1.0 | November 17, 2026

Executive overview

LarkAI brings local artificial intelligence, collaboration across multiple AI providers, communication, translation, browsing, creative tools, project work and app deployment into a single desktop environment. The network connects ideas with the tools, people and services that help develop them into useful results.

LarkAI v1.0 unites Mac and Windows desktop implementations, platform-specific inference resources, member-specific workspaces, encrypted storage, project persistence, communication controls and an application ecosystem. These capabilities work together in a live, integrated platform that includes the digital-asset and marketplace services described below.

LarkAI's differentiation lies in this combination and the way it is organized around the user. Local AI can support private work; optional external models can contribute through reviewed collaboration; projects and creative activity remain organized within the same environment. Communication and app deployment extend the platform beyond an isolated chat session.

LarkAI v1.0 includes LarkAI Chain, LARK Token, integrated wallets and exchange capability. The experience connects AI-assisted work, creator participation and digital-asset services within the LarkAI environment. The desktop environment connects these services through user-controlled workflows across the network.

A connected environment for everyday work

Digital work often spans separate AI accounts, browser sessions, messaging tools, translation services, content applications and development environments. Moving between them adds repeated setup, fragmented context and uncertainty about where information is being processed.

LarkAI addresses that fragmentation through an inclusive product architecture. Users can access several categories of work from one environment while retaining explicit choices about local processing, external services and sharing. The platform can appeal to individual creators, small teams, developers and organizations seeking a coherent entry point into AI-assisted work.

The purpose of the Integrated AI Network is to make these capabilities easier to access and use together. Its value can be assessed through practical outcomes: completing tasks, continuing projects, communicating clearly and participating in the wider ecosystem.

System architecture and operating boundaries

LarkAI separates the user experience from the services that process information, connect peers and settle transactions. This modular architecture allows each function to have a defined permission boundary. It also helps operators assess a component without assuming that trust in one service extends automatically to the rest of the network.

Layer	Technical responsibility	Practical significance
Desktop application	Navigation, authenticated member sessions and OS integration	One environment for related tasks with explicit device permissions
Private workspace	Encrypted local storage, project persistence and member-scoped records	Continuity without automatically uploading private work

Layer	Technical responsibility	Practical significance
AI coordination	Local inference, configured API requests, response screening and synthesis	Choice of models with controlled disclosure and visible dependencies
Peer communication	Verified connections, encrypted text, calls and bounded file transfers	Direct collaboration with distinct transport and message protections
Application ecosystem	Isolated previews, listing metadata, eligibility and entitlements	A path from creation to controlled distribution and commerce
Economic settlement	Wallet authorization, transaction submission and chain verification	Verifiable asset movement without giving AI automatic signing authority

The private processing layer and settlement layer serve different purposes. A conversation can remain local without waiting for blockchain consensus. A token payment requires independently verified ledger state. Keeping these paths separate avoids turning every interaction into a public transaction and prevents an AI response from being mistaken for an authorized financial action.

Mac and Windows download packages

LarkAI v1.0 is distributed through Mac and Windows download packages. The Windows implementation extends access to users and organizations working in Windows environments while preserving the same product concepts: private workspaces, AI choice, projects, communication and explicit transaction authorization.

Shared workflows do not imply identical native code. macOS-specific technologies described in this paper, including Apple CryptoKit, WebKit profiles and macOS quarantine, describe the Mac implementation. Windows requires its own cryptographic implementation, embedded-browser integration, file handling and operating-system permission controls. The availability of a Windows package should not be interpreted as proof that Apple frameworks run on Windows or that every hardware configuration has identical capabilities.

Interoperability depends on protocol compatibility rather than matching interfaces alone. Cross-platform encrypted messaging requires consistent key and ciphertext encodings, canonical signed payloads, suite identifiers, UTF-8 limits and error handling. Independent test vectors should show that each implementation can validate and decrypt messages produced by the other. Invalid signatures, unknown suites and altered authenticated headers must be rejected consistently.

Package documentation should identify supported OS versions, CPU architectures, memory and storage requirements, model availability and update instructions for each platform. Release signatures, published checksums and dependency inventories support provenance checks; device-level validation establishes performance and functional coverage. This paper does not assign unverified Windows minimum specifications or claim a completed cross-platform security certification.

The LarkAI v1.0 network

Capability	What LarkAI brings together	Scope and dependencies
Local AI and voice	Local language and speech resources, conversational assistance and installed Mac speech voices	Performance and inference quality depend on the device and task
Synergize AI	Reviewed collaboration, response comparison and synthesis across configured providers	Optional APIs require user keys, supported model IDs and provider billing
Sentinel AI	Screening, quarantine, provider blocking and collaboration pause controls	Deterministic screening can miss harmful material or flag legitimate text
Endeavour	Persistent project selection, drafts and AI responses in owner-specific storage	Saved outputs do not automatically become executable instructions
Compass	Native browsing, tabs, search choices, session history and decentralized gateway access	Network services and gateway availability remain external dependencies
Translate and Cue Encrypt	Text and speech translation, encrypted drafts and recipient-specific message handoff	Language quality, peer connectivity and key verification matter
Cue and communication services	Verified peer communication, call and file-transfer workflows, plus selected external messaging access	Peer connectivity, device permissions and third-party accounts govern availability
Beacon	Media posts, audience controls and explicit peer sharing	Public audience permission does not mean hosted public internet publication
Guild	Local face-processing workflow with permission attestation	Attestation records the user's declaration; it does not independently verify consent
Agon Ecosystem	Playable games and private deployment of self-contained HTML games and apps	Public discovery, commerce, settlement and entitlements support marketplace participation
Vault and MintAI	Integrated wallets, transaction review, chain activity and participation interfaces	Transactions depend on chain services, user authorization and applicable asset rules

An inclusive workspace does not imply that every third-party service is included in one price. External API usage, accounts, connectivity and licensing can carry separate requirements and costs.

AI choice across the network

LarkAI supports a local-first workflow alongside optional access to OpenAI, Anthropic Claude, Gemini, DeepSeek, Grok and Perplexity provider integrations. Users configure their own API credentials and exact model identifiers for automated API workflows. Website subscriptions and API access remain separate.

Synergize AI gives that choice a practical structure. A reviewed query can be sent to configured APIs; returned material is screened and assessed locally; usable responses can inform a synthesis. A separate workflow supports reviewed website handoffs and user-imported responses.

LarkAI organizes access to different model providers within the same network, giving users a choice of services for different tasks. Comparison scores remain AI-generated assessments, not independently measured accuracy or proof that consensus is correct.

AI request lifecycle and data control

A local request follows the selected local model path. An external request requires the configured provider, an exact model identifier, active optional-service permissions and review of the query to be disclosed. Provider credentials remain member-scoped configuration rather than prompt content. A website handoff follows a separate reviewed copy-and-paste workflow; signing in to a website does not automatically authorize an API request.

Synergize collects usable responses, screens imported content and prepares a comparison or synthesis. A provider failure must remain visible rather than silently becoming a fabricated response. Quarantined material is excluded from synthesis. Local assessment scores describe the supplied answers; they are not calibrated probabilities of correctness. Agreement among models can reflect shared errors, so important conclusions still require source review.

Session boundaries also govern asynchronous work. Locking or changing the active member invalidates late results and stops subsequent requests where cancellation is still possible. An external request already transmitted cannot be recalled. For organizations, the practical control is knowing which content leaves the device, under which account and with whose authorization.

A complete workflow from idea to settlement

Consider a creator developing a multilingual interactive app. Endeavour holds project drafts and responses; local AI helps refine the concept; optional external models contribute only the reviewed material selected for collaboration. Translate prepares language variants, and Cue allows the creator to share an encrypted message or a reviewed file with a verified collaborator.

Agon separates a private preview from a public listing. Listing type, pricing, region and age metadata inform publication and purchase eligibility. A purchase creates a commercial event, while the wallet and settlement services handle the authorized asset transfer. An entitlement represents the buyer's permitted access; it should be issued from verified payment and eligibility state, not a client-supplied balance or screenshot.

This workflow links technical boundaries to practical value: private development reduces unnecessary exposure, reviewed collaboration controls disclosure, isolated previews limit app permissions, and verified settlement supports a clear relationship between payment and access. Refunds, subscriptions and cancellations require corresponding commercial records; a blockchain transfer alone does not define those customer rights.

From conversation to ongoing work

Endeavour supplies continuity through saved projects, drafts and responses. Compass brings browsing into the surrounding workflow. Translate supports multilingual interaction, while Cue gives communication its own verification and sharing controls. Beacon adds audience-aware media sharing.

Together, these functions connect conversation with ongoing work. A user can develop an idea, organize the project, review outside information, translate material and prepare it for sharing within one environment. The value comes from reducing repeated setup and keeping related activity accessible.

LarkAI brings these workflows together in a distinctive platform experience. The network combines continuity between activities with explicit control over local processing, collaboration and sharing.

Privacy and controlled collaboration

LarkAI uses encrypted local workspace storage and authenticated member boundaries. Its external collaboration design preserves explicit review points and restricts the permissions available to model outputs. Sentinel AI screens selected outbound and imported text, quarantines flagged responses and can pause affected collaboration.

External models do not receive direct wallet-signing or native execution authority through Sentinel. These boundaries help users retain control over consequential actions while working with AI.

Cue Encrypt adds a specific encrypted-message workflow using native Apple CryptoKit. The documented construction includes X-Wing ML-KEM-768/X25519, ML-DSA-65 signatures and AES-256-GCM. LarkAI v1.0

includes complete ciphertext copying and accessible key controls. Recipient selection and verified key exchange remain integral to the workflow.

These capabilities support precise security messaging. They do not establish that the whole application is quantum secure, independently audited or immune to compromise. The documented message design lacks forward secrecy and exposes metadata. These boundaries apply to the message design; they should be considered when choosing how to share sensitive information. External website profile storage also has different protection boundaries from the encrypted vault.

Translate encryption architecture

Translate separates language processing from cryptographic protection. Choosing LarkAI as the target creates an encrypted draft bound to the author's identity. Cue Encrypt transfers ciphertext into Cue without automatically sending it. Selecting a verified recipient and sending creates a new envelope addressed to that peer. On receipt, authentication and decryption occur before any requested local translation. Translating text is not itself encryption, and encryption does not verify the accuracy of a translation.

Each member has separate X-Wing ML-KEM-768/X25519 key-encapsulation and ML-DSA-65 signing keys. X-Wing combines a post-quantum component with an elliptic-curve component. The sender encapsulates to the recipient's public key, derives a 256-bit key using HKDF-SHA256 and encrypts the UTF-8 text using AES-256-GCM. A separate encapsulation protects a sender-accessible copy of the data key.

ML-DSA-65 signs the canonical payload, including the encapsulations, ciphertexts and authenticated header. The recipient checks the pinned signing identity and signature before decapsulation; AES-GCM authenticates the encrypted content and header. Version, cryptographic suite, identity, language and size checks reject invalid envelopes. The documented limits are 5,000 UTF-8 bytes for plaintext and 24,000 UTF-8 bytes for an encrypted envelope.

Public message keys are pinned to a member name and SHA-256 fingerprint after verification. Private key seeds remain accessible only to native code and are stored with owner-only file permissions inside the encrypted member vault. They do not enter JavaScript, Python, model context or Cue history. Readdressing a draft occurs within native CryptoKit; plaintext does not enter the Cue composer or an AI service. The local backend rejects routing encrypted-message envelopes into model inference.

The native bridge checks the active member session, vault, local service port, session generation and trusted application frame. Locking, navigation or a changed session invalidates stale responses. A changed peer, connection or edited draft cancels a pending send; duplicate pending sends are blocked. Key import and removal require native confirmation.

These are application-envelope protections, distinct from the WebRTC transport. Static message keys do not provide forward secrecy. The design does not provide message expiry, replay suppression, remote wipe or revocation of plaintext already delivered. A lost identity key can make messages permanently unreadable; a compromised unlocked endpoint or recipient can expose plaintext. Post-quantum primitives do not constitute independent certification of the full application.

Peer to peer communication architecture

Cue connects verified peers through WebRTC rather than requiring every interaction to pass through an AI provider. Pairing and verification establish the peer session; message-key exchange adds a pinned identity boundary for encrypted text. Direct connection availability depends on network reachability. Pairing over a local network or VPN does not by itself guarantee connectivity across the public internet, and any configured hub or relay remains a separate infrastructure dependency.

Audio and video use transceivers prepared on the verified DTLS peer connection. Call identifiers bind ringing, acceptance and termination to the correct call. Camera and microphone capture begin only after acceptance and stop on hangup, lock or reset. Stale permission responses cannot restart an ended call. The native permission bridge is restricted to the trusted local application frame; provider and browsing pages do not inherit that access.

File transfers require the receiver to review the offer. Files are bounded to 50 MB, sent in 16 KiB chunks and controlled by a 1 MiB queued-data limit. Transfers expire after 60 seconds without progress. SHA-256 verification occurs before the authenticated local save operation finalizes a received file. Files are saved with owner-only

permissions, marked with macOS quarantine and not opened automatically. A matching digest verifies integrity, not freedom from malware.

Beacon applies audience checks before preparing peer payloads. Private and selected-recipient sharing limits application-level distribution, while Cue verification governs the connection. A recipient can still retain or independently redistribute delivered material. P2P transport, audience controls and encrypted-message envelopes provide different protections; none should be described as universal anonymity or guaranteed confidentiality on a compromised device.

Agon Ecosystem and creator participation

Agon extends LarkAI into interactive content. LarkAI v1.0 combines playable games, private previews and public game and app deployment through the Deploy New Game workflow in Agon or Endeavour.

Creators can save self-contained HTML snapshots with descriptions, categories, promotional text, placement preferences, country availability, age declarations and content notices. Listing metadata supports free, per-launch, monthly, yearly and perpetual pricing structures. Private snapshots are limited to 500 KB and run through an isolated viewer; native executables are not accepted.

The Agon marketplace connects the creator journey from private preview and commercial listing preparation to public discovery and purchase. Seller review, eligibility verification, checkout, recurring subscriptions, refunds and entitlements support marketplace operations. Publication and purchases depend on verified settlement and eligibility services.

The ecosystem connects people who create apps and games with those who discover and use them. Distribution, payment infrastructure and marketplace operations support that exchange, while transaction completion and clear entitlements help make participation reliable.

LarkAI Chain and LARK Token

LarkAI Chain is the blockchain layer of the LarkAI v1.0 ecosystem, with LARK Token as its native digital asset. Together, they connect ecosystem activity with verifiable transactions and digital-asset services within the platform.

LarkAI Chain and LARK Token support ecosystem payments, creator settlement and participation mechanisms under the network's operating rules. Detailed token utility, issuance, supply, allocation, vesting, fees, governance and reward parameters belong in the accompanying network and token specifications. This overview does not set or amend those terms.

LARK Token should be evaluated through documented utility and actual network activity. This whitepaper makes no promise of price appreciation, investment return, external exchange listing or guaranteed liquidity.

Decentralized blockchain architecture

LarkAI Chain's architectural objective is decentralized settlement: participating nodes independently verify transactions and maintain a shared ledger rather than treating one application database as the authority for token ownership. This describes the network design; the extent of decentralization must be established by its deployed node operators, consensus rules and control structure.

Transaction authorization belongs to the wallet holder. AI-generated suggestions are inputs for review and do not confer signing authority. A wallet operation must identify the network, asset, recipient, amount and applicable fees before authorization. The chain's accepted transaction state determines settlement; an application display or local receipt alone is not proof of an on-chain transfer.

Private content and public settlement occupy separate layers. Messages, translation text, prompts, local projects and private keys should remain off-chain; a public ledger can expose addresses, amounts, timing and interaction patterns. Encryption protects selected content but does not make public transactions anonymous. Publishing a content hash can also create a lasting correlation, so only deliberately selected settlement data should be committed to the ledger.

A complete chain specification must identify consensus and finality rules, validator or node admission, network identifiers, fees, issuance, contract execution, upgrade authority and recovery procedures. This paper does not assign an unverified consensus algorithm, transaction rate, finality time, validator count or decentralization score. Independent operators, verifiable ledger access and transparent administrative powers are the evidence needed to substantiate decentralization in operation.

LARK Token utility and ecosystem value

LARK Token connects the platform's digital-asset services with its wider community. Its practical value proposition is access to defined economic activity: supported ecosystem payments, creator settlement and eligible participation mechanisms. It does not grant ownership of private user data, automatic access to encrypted messages or an unconditional claim on platform revenue.

Area	Value to participants	Relationship to LARK Token
Everyday use	AI, translation, communication and projects in a connected environment	Supported purchases can create transaction utility; local private work need not become an on-chain transaction
Creators and developers	App distribution, pricing options and a route to settlement	Eligible marketplace purchases and creator payments provide service-linked usage
Community participation	Contribution to useful content, services and network activity	Rewards apply only where published eligibility and verification rules authorize them
Network operation	Verifiable settlement and transparent transaction state	Fees and incentives must follow the chain's actual rules; staking or governance rights are not assumed
Wallet and exchange users	Authorized transfers and supported asset exchange	LARK can participate in supported asset flows, subject to fees, liquidity and service coverage
Privacy	Separation of private content from economic settlement	Possessing tokens does not override encryption, consent or access controls

The platform benefits when token use solves a practical problem for participants. Creators need dependable settlement, users need clear costs and useful services, and the community needs understandable participation rules. Requiring token purchases for actions that do not need settlement can add friction; utility should justify that requirement wherever it applies.

Adoption driven demand and market value

LARK's economic thesis connects useful services to repeat activity. More useful apps and services can attract users; satisfied users can return; repeated purchases and settlement can increase demand for the token where LARK is required. This is a potential demand mechanism, not a mechanical relationship between user growth and token price.

Market value also depends on circulating supply, issuance, unlocks, participant selling, liquidity and how quickly tokens are spent and reused. Immediate conversion by sellers or rapid token circulation can offset increased transaction demand. Platform growth can therefore coexist with a flat or falling token price. No price floor, natural appreciation, guaranteed return or superior performance relative to other assets follows from the product's feature set.

The meaningful distinction from a token project driven primarily by promotion is the opportunity to measure actual utility. Relevant indicators include active paying users, repeat buyers, creator retention, completed settlements, token-denominated service volume, fee burden, token concentration and liquidity quality. Trading volume alone does not demonstrate demand for the platform's services.

A durable ecosystem should make these indicators observable alongside supply and allocation rules. Any burn, staking, governance, buyback or revenue-sharing mechanism must be supported by published token terms and deployed implementation before it is claimed. None is presumed in this paper.

A distinctive integration of AI and tokenization

LarkAI combines private local AI workflows, multi-provider collaboration, translation encryption, verified P2P communication, creator deployment and blockchain-linked economic services in one user environment. Its distinctive proposition is the continuity between these activities and the separation of private content from authorized settlement.

Existing decentralized AI ecosystems demonstrate the importance of defined protocols and economic coordination. Bittensor focuses on a peer-to-peer intelligence market; Fetch.ai describes a layered architecture for agent identity, communication and settlement. LarkAI's focus is the user-facing integration of private work, encrypted communication, creator activity and authorized digital-asset services. Its differentiation rests on how these workflows function together.

Wallets and exchange capability

Vault is the integrated wallet interface for the v1.0 ecosystem. It provides receiving addresses, verified balances and transaction history, transfer preparation, fee review and explicit transaction authorization. Asset and network identities must be clear throughout each transaction.

Smart-contract workflows connect implementation planning with reviewed artifacts and authorized deployment. Supported chains, contract identities, signing arrangements and transaction outcomes define the boundaries of each deployment.

Integrated exchange capability provides access to supported asset exchange within the ecosystem, with visible pricing, fees, authorization and settlement status. Service documentation identifies supported assets, the execution model, custody arrangements, liquidity sources and operating responsibilities. Inclusion in the platform does not guarantee liquidity or availability in every market.

MintAI provides visibility into eligible network participation and rewards under published verification and issuance rules. Chat activity and game scores earn no token entitlement unless the network's operating rules expressly make those activities eligible and verify them.

Access and participation

LarkAI brings personal work, shared communication, content creation and digital-asset services into an integrated network. People can use its local capabilities, connect optional AI services, organize projects, share with selected peers and explore apps and games through Agon.

Access arrangements can include desktop editions or subscriptions, organizational deployments, creator services and supported digital-asset services. Pricing and service terms define what is included, which functions require external accounts and which activities carry additional charges.

A complete user journey can connect local assistance, reviewed multi-provider collaboration, project continuity, translation, encrypted messaging and app deployment. Supported wallet transactions and exchange services extend that journey into the digital-asset ecosystem.

Network development and participation

The network develops through contributions from users, creators, developers and service providers. Feedback on usability, accessibility, language quality and reliability helps guide improvements. Developers and creators can contribute useful apps and content, while infrastructure providers can support model access, payments and network services.

Blockchain, wallet and exchange integrations require clear operating responsibilities, verifiable transaction flows and explicit authorization. Service documentation describes service availability and the protections that apply to each activity.

Reliability and evaluation framework

A credible network can be evaluated through observable behavior as well as its feature list. AI evaluation should measure task completion, response usefulness, error rates and latency for specified models and devices. Encryption evaluation should include known-answer tests, altered-envelope rejection, identity verification and key-loss recovery behavior. Peer testing should include interrupted calls, expired transfers and mismatched file digests.

Economic services require a separate evidence trail: authorized transaction, accepted network state, settlement confirmation and the resulting entitlement or receipt. Duplicate submissions, delayed finality, unavailable infrastructure and incomplete settlement must have explicit outcomes. Retry behavior must not cause duplicate financial effects. These are acceptance criteria for assessing the integration, not claims of measured performance results.

Capacity planning should distinguish local compute from shared services. Local inference depends on model size and device resources; external inference depends on provider limits and costs; peer calls depend on endpoint and network conditions; settlement depends on chain capacity and finality. No single throughput number meaningfully describes all four paths.

Published operating metrics should pair adoption with service quality. Repeat task completion, active creators, completed purchases, payment failures, support resolution and infrastructure concentration offer a more useful picture than token trading volume alone. Benchmarks should identify their method, sample, environment and limitations before they are used in commercial claims.

Live network and service availability

Released in November 2026, LarkAI v1.0 brings the Integrated AI Network into operation. Local AI, multi-provider collaboration, communication, translation, creative tools, project work and app deployment are connected with marketplace and digital-asset services in one environment.

Users can work locally, connect supported external services and participate in the ecosystem through the features available to their account and region. App publication, payments, wallet transactions and exchange settlement remain subject to the relevant eligibility checks, user authorization and service terms.

Network availability depends on more than the desktop application. External providers, peer connectivity, chain infrastructure and settlement services each have their own operating conditions. Product documentation and service notices identify supported environments, coverage and maintenance information.

The Integrated AI Network

LarkAI brings assistance, collaboration, communication, creative activity and app deployment together in a shared product environment. Its purpose is to help people move from a question or idea to practical work, ongoing projects and participation in a broader ecosystem.

LarkAI v1.0 connects these capabilities with LarkAI Chain, LARK Token, wallets and exchange services. A consistent experience, clear user controls and verified technical delivery are central to making the network useful and dependable.

Product and release specifications

This whitepaper describes the LarkAI v1.0 Integrated AI Network. Accompanying product and service specifications cover supported models and devices, security boundaries, chain architecture, token parameters, wallet authorization, exchange operation and marketplace service terms.

External accounts, APIs and networks retain their own configuration, pricing and availability requirements. Feature access depends on the supported device, account configuration, service coverage and applicable market restrictions.

Technical and ecosystem references

- Apple CryptoKit documentation: <https://developer.apple.com/documentation/cryptokit>
- NIST ML-KEM standard FIPS 203: <https://csrc.nist.gov/pubs/fips/203/final>
- NIST ML-DSA standard FIPS 204: <https://csrc.nist.gov/pubs/fips/204/final>
- Bittensor protocol research: <https://www.bittensor.com/academia>
- Fetch.ai token utility documentation: <https://network.fetch.ai/docs/guides/network/intro-fet-token>
- Fetch.ai multi-agent architecture: <https://innovationlab.fetch.ai/resources/docs/intro>

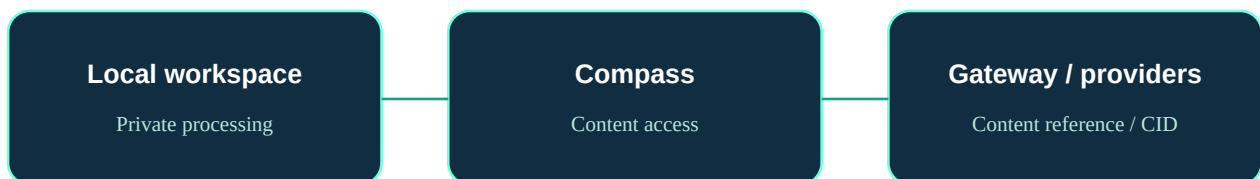
Cryptographic specifications describe primitives, not certification of LarkAI. References to other ecosystems establish relevant technical precedents and do not imply affiliation or endorsement.

- Fetch.ai architecture paper: <https://arxiv.org/abs/2510.18699>
- Bittensor whitepaper: <https://www.bittensor.com/whitepaper>

Decentralized Internet

LarkAI v1.0 | Whitepaper extension

This extension accompanies the supplied whitepaper. Original pages are retained without alteration. It clarifies content access and the boundaries between distributed delivery, peer communication and settlement.



Cue peers and authorized chain settlement remain separate layers.

Access, integrity and user control

LarkAI's Decentralized Internet vision connects the workspace to information and services distributed across independently operated systems. Compass is the user-facing entry point for regular web browsing and decentralized gateway access. This extends how users discover and retrieve content; it does not imply that LarkAI owns the internet or operates every node and gateway.

Content-addressed networks such as IPFS identify data by a content identifier (CID), rather than only by the location of a hosting server. This gives publishers and readers a stable reference to a particular content version. A CID can support integrity checking against that reference; it does not certify that a publisher is trustworthy or that a document is accurate. [1]

The intended workflow is straightforward: select a resource in Compass, resolve its content reference through the configured access path, retrieve the available content and review it within the browsing boundary. An HTTP gateway provides a bridge for conventional web clients; the gateway remains an infrastructure dependency, and availability depends on reachable providers. Native peer retrieval and independently operated gateways should be identified separately in product specifications. [2]

For creators, this architecture offers a path to versioned publications, reference libraries and distributed delivery of public materials. Multiple providers can reduce reliance on one hosting location when the same content is maintained and reachable. Persistence requires an explicit retention or pinning arrangement; a content identifier alone is not a promise of permanent storage. The network's operating plan should identify who retains data, who pays and how availability is monitored. [3]

Decentralized delivery does not mean private delivery. IPFS content and routing metadata may be observable to other participants or gateways. Private prompts, notes, projects, identity keys and unencrypted messages should not be published automatically. Cue Encrypt protects selected message content under its own recipient and key-verification model; it is separate from browser access and does not make public browsing anonymous. [4]

Content distribution, peer communication and financial settlement serve different purposes. Compass retrieves information; Cue connects verified peers; LarkAI Chain records authorized settlement where applicable. Reading a page need not create a token transaction, and owning LARK does not grant access to someone else's encrypted workspace. Optional payment and entitlement flows must remain explicit.

The deployment specification should state supported URI schemes, gateway choices, content-verification behavior, isolation of untrusted pages, pinning and retention responsibilities, abuse handling and fallback behavior. Native IPFS hosting, automatic replication, universal censorship resistance, guaranteed uptime and anonymous routing are not established by the supplied description of gateway access. These are separate capabilities to document and validate before they are claimed.

Decentralized Internet references

[1] IPFS - How IPFS works: <https://docs.ipfs.tech/concepts/how-ipfs-works/>

[2] IPFS Gateway: <https://docs.ipfs.tech/concepts/ipfs-gateway/>

[3] IPFS Persistence: <https://docs.ipfs.tech/concepts/persistence/>

[4] IPFS Privacy and encryption: <https://docs.ipfs.tech/concepts/privacy-and-encryption/>

LarkAI Security Overview

Blockchain and crypto architecture • Encrypted Cue • Wallet • AI controls

LarkAI’s security model combines encrypted local storage, verified peer connections and strict boundaries around AI collaboration. Its strongest documented protections keep private keys in native code, require deliberate sharing and prevent AI responses from becoming executable instructions. These controls support privacy and user independence across the platform. [1–3]

Scope and assurance

This overview describes the supplied 0.47 source package and its accumulated security documentation. “Documented” means described in those materials, not independently audited or tested for this report. Layer 0 and Layer 1 below are proposed architectural roles: the materials do not establish a deployed LARK consensus network, validator set or live token settlement. [1–3]

Security layers across LarkAI

Layer	Security type and boundary	Status
Device and identity	Mac authentication, member roles, session revocation and native origin checks	Documented
Storage and recovery	AES-256 encrypted APFS vault; optional offline recovery envelope	Documented
Communication	Verified Cue peer transport; separate signed message encryption in Translate	Documented
AI collaboration	Reviewed sharing, constrained provider APIs and untrusted text output	Documented
Sentinel AI	Pattern screening, quarantine, integration blocking and local containment	Documented
Wallet and transactions	Disconnected wallet UI; future explicit signing and verified chain evidence	Backend pending
LARK network	Proposed Layer 0 connectivity and Layer 1 validation and settlement	Not established

Meaning of security levels

Cryptographic types describe specific mechanisms, such as AES-256-GCM encryption. Operational levels describe escalating local responses, such as quarantine or suspension. Neither is an overall security score. Using NIST-standard algorithms does not establish FIPS module validation, military certification or complete post-quantum protection for LarkAI. [1,2,4,5]

LARK Blockchain Crypto & Wallet Security

Layer 0 proposed network foundation

For this draft, Layer 0 means the proposed network substrate for node connectivity and any future communication between chains. Its security design should specify peer identity, message authentication, rate limits, resistance to malicious peer concentration and denial of service, and separation of node services from private AI workspaces. Cross-chain functions would need their own proof verification and trust model. These are design requirements, not verified LARK capabilities.

Layer 1 proposed blockchain and settlement

Layer 1 means the proposed base ledger that validates transactions and establishes the accepted state. Before describing it as an operational decentralized blockchain, LARK needs a published consensus mechanism, validator admission rules, finality assumptions, transaction signature scheme, replay protection, state-transition rules and upgrade governance. Independent node operation and the distribution of validation authority must substantiate decentralization. P2P connectivity alone does not establish ledger consensus.

A credible security case must explain how the ledger handles double spending, conflicting histories, validator collusion, network partitions and unavailable nodes. If bridges, contracts or oracles are introduced, their permissions and failure modes need separate review. The supplied materials do not substantiate live mining, token rewards, deployed contracts or a production settlement network. [1,3]

Wallet security current boundary

Vault currently provides a disconnected wallet integration workspace. The documented UI has no configured chain, contract address or signer. Balances and addresses remain unavailable, transfer previews do not send, and contract briefs do not execute or deploy. The UI requests no wallet seed phrase or private key. Local JSON exports are metadata, not wallet-key backups. [3]

Controls required for a live wallet

A future wallet should isolate signing keys from AI and web content, use an explicitly selected custody model and require user approval of the actual transaction. The confirmation should bind chain, recipient, asset, amount, fees and any contract permission. Signing must use chain-specific validation, replay protection and verified network evidence. Hardware-backed signing, multisignature treasury controls and spending limits are recommended options, not documented features.

The backend must verify balances and transaction status against chain evidence. Explorer hash-format checks cannot prove a transaction exists. Wallet recovery must be designed and tested separately from vault recovery and Cue message-key backup. No autonomous AI signing or unrestricted transaction authority should cross this boundary. [1,3]

Translate Encryption & Encrypted Cue

English to Encryption message flow

Translate accepts English or another source language and produces a LarkAI encrypted message. The initial draft encrypts to the sender's own key. Cue Encrypt places ciphertext in Cue without sending. The sender selects a recipient, verifies the Cue connection and confirms message-key exchange. Native code then authenticates and re-encrypts the draft for that recipient before sending. [2]

The recipient explicitly opens the ciphertext in Translate. Native code verifies identity and signature, decrypts the message and performs local language translation only if required. Cue history retains ciphertext. Plaintext does not enter the Cue composer or AI inference in this documented flow. Ordinary language translation and cryptographic encryption are distinct operations. [2]

Encryption types and key protection

Control	Documented construction
Shared-secret establishment	X-Wing hybrid ML-KEM-768 and X25519
Key derivation	HKDF-SHA256 derives a 256-bit encryption key
Message confidentiality and integrity	AES-256-GCM encrypts text and authenticates the header
Sender authentication	ML-DSA-65 signs the canonical payload; native code verifies before decryption
Key storage and trust	Separate member keys in the encrypted vault, mode 0600; native-only seed access; pinned public keys and SHA-256 fingerprints
Session enforcement	Trusted loopback main frame, active member and vault checks; stale operations discarded after session changes

Scope of post quantum protection

ML-KEM and ML-DSA are standardized by NIST for key establishment and digital signatures. Their use here applies to the documented message envelope. It does not upgrade Cue's underlying WebRTC transport, third-party TLS, the operating system or a future blockchain. The envelope remains a new, unaudited application protocol requiring independent review and live two-device validation. [2,4,5]

Limits that matter

Static sender and recipient keys provide no forward secrecy: later key compromise can expose recorded envelopes. Names, key identifiers, original language and message IDs remain visible. The documented protocol has no replay suppression, message expiry or remote wipe. Recipients can copy plaintext, and an unlocked compromised device can expose it. Losing message keys and their protected backups can permanently lose access. [2]

Compass Decentralized Internet Security

Decentralized content access

Compass provides a documented decentralized browsing mode for IPFS and IPNS content. It maps IPFS CIDv1 identifiers and IPNS base36 keys to separate HTTPS subdomains on inbrowser.link, the public IPFS service-worker gateway. This extends LarkAI's decentralized access model while keeping external content separate from the trusted application. It does not establish a LARK blockchain node, validator or Layer 0 consensus service. [1]

Security types and enforcement boundaries

Security type	Documented browser control
Origin isolation	Separate gateway subdomains isolate content origins rather than placing unrelated content under one shared browser origin
Transport and navigation	Decentralized mode rejects HTTP, embedded URL credentials, local files, unexpected ports and foreign top-level origins; no TLS bypass is installed
Application privilege separation	Gateway and provider pages have no LarkAI native tool bridge; remote content cannot use that bridge to operate the local workspace
Device permission restriction	Camera and microphone requests are limited to the exact trusted loopback main frame; Compass and provider pages are denied
Session and history	Ordinary Compass uses a nonpersistent browser store; its bounded in-memory history clears with the browser pane at lock or Clear session
Local containment	The native emergency action closes Compass and provider views; it does not disable the remote site or recall information already sent

Privacy and external dependencies

Decentralized access still depends on gateway bootstrap scripts, delegated routing, content providers and optional relays. Availability and privacy therefore depend partly on external infrastructure. HTTPS and origin isolation do not provide anonymity, hide every network request or guarantee that content remains available. Ordinary websites do not become decentralized merely because Compass opens them. [1]

The nonpersistent Compass boundary differs from embedded provider sessions. Provider website profiles persist separately for each member and service in app-scoped WebKit storage outside the encrypted vault. They rely on operating-system protection, including FileVault at rest, and remain subject to provider expiry and revocation policies. Clearing Compass history does not erase provider records, external logs or downloaded exports. [1]

How Compass fits the platform security layers

Compass adds browser-origin separation and restricted navigation to the platform's device, identity and application controls. Sentinel screens selected AI collaboration exchanges, but the source does not establish comprehensive scanning of every website or download. Browser content remains untrusted. The supplied documentation records limited native WebKit retrieval tests, not an independent browser security audit or proof of universal network reachability. [1]

Synergize Sentinel AI & Platform Enforcement

Synergize AI security

Automatic collaboration requires exact-query review, configured provider credentials and optional-service authorization. Six fixed HTTPS endpoints reject redirects and environment proxies, and responses are bounded. API keys stay in owner-scoped files inside the encrypted workspace and are excluded from prompts and response JSON. Website handoffs require review and manual pasting. No private library attaches automatically. [1]

Sentinel screens outgoing and incoming text. Quarantined responses stay out of synthesis. AI outputs remain untrusted text, with no direct filesystem, tool, application-control or wallet-signing privileges. Lock cancels subsequent work and discards stale results, but cannot recall a request already sent to a provider. Provider retention and billing remain external dependencies; agreement among models does not establish correctness. [1]

Sentinel AI as police for AI agents

The police analogy describes local rule enforcement: inspect selected exchanges, flag prohibited patterns, isolate suspicious responses and require human review. Sentinel is documented as a deterministic defensive boundary, not a trained threat detector. It checks selected credential, override, exfiltration and destructive-command patterns. “Military police” is an analogy only, not a military assurance level or authority over remote AI systems. [1]

Response level used in this overview	Control type and action
1 Screen	Preventive and detective: inspect selected outgoing and incoming text
2 Contain	Quarantine flagged responses, exclude them from synthesis and block that provider integration per member
3 Suspend	Pause new collaboration and invalidate pending comparisons
4 Emergency containment	Native action closes provider and Compass views; pending peer AI assistance is cleared; restoration requires appropriate review

These four labels organize documented controls for this overview; they are not shipped certification tiers. Filters can miss attacks and flag benign text. Sentinel can revoke local access and contain integrations, but cannot shut down or retaliate against a remote model. [1]

Platform boundaries and evidence

The encrypted APFS vault protects locked local data. Mac authentication, a separate vault passphrase and session controls govern access. Optional offline recovery uses a hybrid envelope. External WebKit provider profiles sit outside the vault and rely on OS protection such as FileVault. Forced termination may leave a vault mounted. The inspected build is ad-hoc signed, without Developer ID notarization or a complete app-wide sandbox. [1]

Sources: [1] app/SECURITY.md; [2] SEALED-MESSAGES.md and app/saint/sealed-text.js; [3] app/WALLET-BACKEND.md, all in LarkAI-0.47-Verified-Source. Earlier security notes are cumulative; the sealed-message document narrows older post-quantum scope statements. This is a documentation review, not a code audit.

Standards: [4] NIST FIPS 203, ML-KEM, <https://csrc.nist.gov/pubs/fips/203/final>. [5] NIST FIPS 204, ML-DSA, <https://csrc.nist.gov/pubs/fips/204/final>. Algorithm standards do not certify this implementation.