JLINX White Paper

September 30, 2022

Jim Fournier, Victor Grey, Jared Grippe

Abstract

JLINX is an automated data agreement layer for the distributed web. A *JLINX Agent* is an online actor that can be granted revocable rights to act on behalf of a specific party. People and organizations are each represented by their own agent as a node in the network, that can sign and hold data agreements with other parties and prove the validity of the data agreements that they hold, without requiring consensus on any global ledger. JLINX agents can record immutable events on any directed acyclic graph (DAG) data structure such as Ceramic, Hypercore or Holochain to reference data, identity and agreements in a fast, truly decentralized network without gas fees, and can also write any event or roll-up to any other ledger or blockchain.

Background

Web3 is a yet-to-be-well-defined term for a collection of technologies that use cryptographic keys and hashes combined with distributed ledgers to achieve decentralized consensus and cross-app cooperation on the internet. In practice, this has largely meant monetized activity on blockchains. This has created a new financial capital market on the internet. However, the consensus methods necessary for ensuring secure financial transactions are inherently too slow and expensive (computationally, financially, and often environmentally) for the every-day high-speed data exchange needed for the decentralized web.

Decentralized

There are a number of possible meanings of “decentralized”. For blockchain, it means a set of separate nodes all running the same protocol to create a single "source of truth", e.g. The Ledger. For dApps and web3 the term decentralized is also often conflated with a consensus algorithm, the process by which “individual action” is turned into a “shared state”. However, turning decentralized action into a centralized state using a consensus algorithm introduces a bottleneck.

Consensus mechanisms

Cryptocurrencies must form consensus from a constellation of disparately owned-and-operated, untrusted nodes each complying with a protocol before appending to a global ledger that produces a single centralized source of truth. That ledger is gated by a monetary consensus algorithm (and often monetary fees). To ensure that no one actor can manipulate the ledger, a complex process of consensus (mining, staking, etc.) must be completed before a transaction can be complete. This results in a slow and costly infrastructure. For web3 to scale for data we need to decentralize the ledger itself.

Servers, services & nodes

The architecture of a decentralized network should support the needs of the parties in that network. Virtually none of the parties on the internet are running servers that create nodes on blockchains, in fact very few are running servers of any kind. Most are using services, where they have a client application that is dependent on the terms dictated by those services. This is increasingly true even for developers and enterprises. The network is made up of cloud services where the activity is all at the API level between those virtual servers. The existence of blockchains is largely orthogonal to the data flows and agreements between parties which are all happening off-chain using proprietary APIs.

Cryptographic standards

For the most part, non-monetary data, permission and meta-data has been outside the scope of web3. And yet the existence of ubiquitous cryptographic key technology (PKI), irreversible hashes, and W3C standards such as decentralized identifiers (DIDs) and verifiable credentials (VCs), all combined with directed acyclic graph (DAG) technology, which could also be described as 'tokenless' distributed ledger technologies (DLTs), make it possible to create a protocol for off-chain data exchange control.

Safely sharing data

Sharing data between dApps is core to the web3 revolution. As more and more apps have access to data about you, having a layer that defines how that data is allowed to be used is critical. Being able to define your terms, view your counter-party's terms, and sign compatible data sharing agreements automatically will significantly increase the security of the distributed web.

While a blockchain sounds like a good place to store these agreements there are better options. DAGs provide the same reliability and verifiability of a blockchain but without the overhead delay or cost of consensus fees.

JLINX

JLINX is a decentralized off-chain data agreement protocol. It gives both people and organizations control over data sharing and decision making. Each party is represented by their own *agent node*. Human and machine-readable contracts provide data control, legal agreements and governance.

JLINX is based on the JLINC Protocol[1] for data control, combined with tokenless *micro-ledgers* built on Ceramic, Hypercore, Holochain, or other DAG based distributed databases. JLINX agent nodes represent each party. Agents can act on your behalf, including by signing contracts, which contain the human readable text, pointers to text, and context necessary to automate exchange

[1] https://jlinc.org/

between data stores. Agents are granted rights by you, and then use their own identity (DID) to write to their own event stream and documents (e.g. Ceramic *Tile or Hypercore)*, to provide non-repudiation and auditable records. Agents can also optionally records events and documents to any combination of other ledgers and blockchains.

> Under Web 2 you gave your data to some website, they stored it in their database, and you hope for the best. With JLINX you control your data, who can see it, what they can do with it, if they're allowed to keep a copy, etc. And everything is logged on immutable ledgers without gas fees.
>
> JLINX is a new data-exchange control layer for web3 dApps that doesn't require you to have a crypto wallet or pay data-exchange fees.
>
> Businesses that want access to data that you might share with them run their own JLINX agent nodes. You, or your agent, can grant those nodes tightly defined access to your data. In the future they might even pay you for their use of your first-party data.

JLINX offers a simple, elegant, cryptographically secure way of signing and exchanging human-readable agreements governing data used across any combination of websites, databases, applications and codebases.

JLINC protocol

JLINX extends the JLINC protocol, which provides a way to assemble methods, systems and techniques to provide confidentiality and to represent fiduciary relationships and accountability for parties sharing data over the internet. It facilitates control over shared data, provenance of that data, and non-repudiation of data sharing actions using standard public/private key cryptography (PKI), JSON linked data (JSON-LD), decentralized identifiers (DIDs)[2] and verifiable credentials (VCs)[3].

JLINX micro-ledgers

For dApp interoperability and data provenance, "micro ledgers" are like individual blockchains where only the owner/key holder(s) can write to their unique ledger address. This provides free, tokenless, immutable, non-monetary data stores where each party writes to its own ledger stream.

Micro-ledgers are single-author topical event streams (tiny blockchain). Each actor, human or code, can create a micro-ledger simply by generating a cryptographic signing key pair, and then writing a series of events (blocks) that are cryptographically signed to ensure order. The public key is considered the ID of the micro-ledger. Two or more actors can create pairs of event streams to form bi-directional relationships with each other. Encryption can be used to form

[2] https://w3c.github.io/did-core/

[3] https://www.w3.org/TR/vc-data-model/

private relationships. DIDs and DID Documents are used to define ownership and access control over the micro-ledgers.

JLINX contracts

JLINX contracts are different from “smart contracts” in two fundamental ways. 1) They are contracts, not computer code, i.e. they can contain the human-readable text of actual legally binding contracts, which specify the intent of the parties in the event of any unexpected behavior by the code. 2) As such, they can ultimately be adjudicated by a judge under the legal system rather than by a compiler.

JLINX contracts are human and machine-readable agreements expressed in JSON and signed with cryptographic keys using standard PKI. This allows human readable agreements, that dictate how data can be used or shared, to be handled automatically and transparently. The expected outcome of the agreement remains apparent to anyone, including a human judge. The parties of the agreement are identified using DIDs and their signatures are executed with standard cryptographic keys using standard PKI.

Any person or any organization of any kind, can use JLINX to enter into secure, irrefutable and legally adjudicable agreements governing the exchange and use of data shared under that agreement. Data sharing agreements can be used stipulate terms such as:

- How long you can store the data
- In what ways you can use the data
- Under what conditions can the data be used, or no longer used
- Under what conditions (time or otherwise) the data should be purged
- Requirements to log and report all usage or sharing of the data
- Requirements to be financially compensated for specific usages of the data

The agreements are created, exchanged and persisted in systems independent of the mechanism used to provide verification and proof of the agreement. This means either party can store copies of the agreement anywhere and in any way they like, and it can still be verifiable by a 3rd party. While agreements might stipulate financial compensation, the agreements and verification of those agreements are independent of any specific system. While JLINX Agents and agreements thrive on Web3 they do not depend on it. JLINX Agents and agreements are independent of persistence and data exchange layers. JLINX Agents could be implemented on a carrier pigeon network.

JLINX agent nodes

Each party to a JLINX contract is represented by their own online “agent” that can manage their signed agreements and hold keys to sign those agreements based on signed instructions from the party represented by that agent.

An agent node is code that runs in the cloud that can be granted permission to take actions on behalf of another entity. Each agent node has its own identifier (DID) and must be given explicit rights to act on behalf of another identifier. These rights are represented as verifiable claims that are immutably written to a micro-ledger and can only be revoked via explicit action by the controlling entity.

JLINX nodes need to be always online to manage data sharing agreements. This requires a private server or a trusted SaaS provider to represent each party online. This ensures that data is always available to other nodes in the network. Agents verify and react to incoming events and requests from other nodes. Like smart contracts, agent nodes can be instructed to take actions (execute code) in reaction to other events on the network. Unlike smart contracts, code is solely executed by an agent representing one of the parties, and results in writing new events on micro ledgers that are owned and operated on behalf of that party.

JLINX Agreements lay the foundation for an automated and decentralized basis for trust. This layer gives everyone greater control over the data they share online. It literally gives each party "agency" in a system based on symmetrical permissioned data exchange. The network architecture maps decentralization to the social, legal and commercial structure of society, where each party not only has "self-sovereign identity", but more importantly "self-sovereign agency."

JLINX Agent as a Service

JAaaS directly addresses the "nobody wants to (or should have to) run their own server" problem articulated by Moxie Marlinspike in his influential post, My First Impressions of Web3[4].

JLINX Agents as a service are hosted and manages by third parties. While you need to trust one of those third parties to not steal your data, The JLINX Protocols allow the root identity owner to revoke rights granted to keys held by any agent. The root identity keys are listed as the "Owner and controller" and your agent's keys are listed as an "Agent Controller" on any documents or event streams that you grant them control over.

Any root identity can grant rights to a hosted JLINX Agent service and later revoke those rights and move those responsibilities to another agent. This allows the development of desktop apps, mobile apps and browser extensions, that can hold and use your subordinate identity keys to interact to multiple JLINX agents in the cloud under the control of your client app that holds your root keys on a local device.

Client apps

Applications on the local devices can use whatever combination of biometric, pin, password or second factor they need. Desktop and mobile apps can take advantage of any KPI storage technology such as Apple's KeyChain, FIDO, RSA Hardware, etc.

[4] https://moxie.org/2022/01/07/web3-first-impressions.html

A hardware key could be used to initially establish a DID and then grant revokable access to DIDs that represent your local apps and online agents, so no matter how many other devices are compromised you have a path to restore full control over your online identity.

Authentication between a hosted JLINX Agent and you (or your local apps) can be implemented using any existing web authentication technology. Hosted agents can also provide data recovery and key recovery.

Decentralized identifiers

Each "actor" is represented using a unique decentralized identifier (DID). Every DID is tied to cryptographic key pairs used to sign any actions taken by that actor. This gives individuals, organizations and software applications persistent self-sovereign identity online. Additional identities (DID-based identifiers) can be created at-will where "persistent pseudonymous identity" with specific parties is required.

For example users (or their agent) can create a new DID for each website they sign up with to ensure those apps cannot cross correlate the activity and data for that individual.

Verifiable credentials

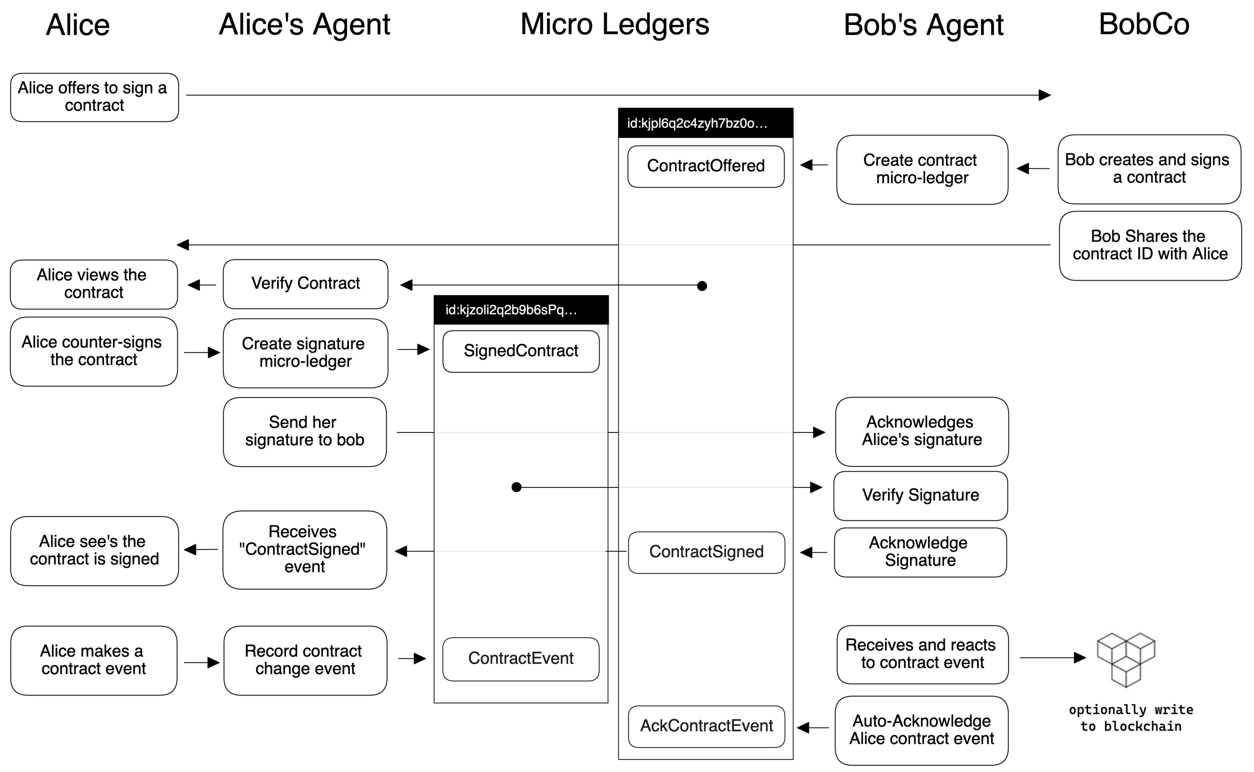
This provides the basis for a functional multi-party implementation of verifiable credentials where each party maintains their own DIDs controlled by their own agent and anchored on their own micro-ledger, which has significant privacy advantages and scales better than verifiable credentials on a cryptocurrency.

Micro-ledger protocols

JLINX is based on a set of distributed protocols for establishing and exchanging self-sovereign identity, agreements, permissions, roles and governance. It uses human and machine readable append-only documents. JLINX Documents (Identity, Profile, Contract, etc.) are all implemented using immutable event streams that serve as an immutable audit trail. A hash of the state of any document can be written to any blockchain to enable data exchange with smart contracts. This allows governance to happen at high-speed (off-chain) but to also interoperate with on-chain smart contracts providing legal documentation and governance for Web3.

JLINX micro-ledgers are currently implemented using the Ceramic protocol where streams called *Tiles* are used to represent and manage immutable multi-party state. The first JLINX demo was built on Hypercore.

Contract flow diagram


Alice
Alice's Agent
Micro Ledgers
Bob's Agent
BobCo
Alice offers to sign a contract
id:kjpl6q2c4zyh7bz0o…
ContractOffered
Create contract micro-ledger
Bob creates and signs a contract
Bob Shares the contract ID with Alice
Alice views the contract
Verify Contract
id:kjzoli2q2b9b6sPq…
Alice counter-signs the contract
Create signature micro-ledger
SignedContract
Send her signature to bob
Acknowledges Alice's signature
Verify Signature
Alice see's the contract is signed
Receives "ContractSigned" event
ContractSigned
Acknowledge Signature
Alice makes a contract event
Record contract change event
ContractEvent
Receives and reacts to contract event
optionally write to blockchain
AckContractEvent
Auto-Acknowledge Alice contract event

- Alice wants to sign a contract with Bob
- Bob creates and signs a contract offering using his DID
    - Bob’s Agent creates a new micro-ledger for his contract events
    - The first event is an event offering that contains either a link to or the contract details themselves.
- Alice’s Agent verifies the contract offering is from Bob
- Alice signs the contract
    - Alice’s Agent creates a new micro-ledger for her contract events
- Bob’s Agent receives the signature, verifies it and writes an acknowledgement event to Bob’s ledger
- Alice’s Agent sees the signature acknowledged event
- Alice now sees the contract as signed using her signature
- Alice’s Agent and Bob’s agent can now communicate about this contract automatically by appended events to their personal ledgers
- Alice then takes some action under the details of the contract
    - Alice’s agent records a “contract event” to her micro-ledger
- Bob’s agent receives the data change event from Alice’s ledger
    - Bob’s Agent can take any programmable actions needed in response to Alice’s event including write a universally unique reference to this event to any blockchain

The events of multiple micro-ledgers are joined, ordered and projected, using shared open source software, to create shared state.

Applications using these protocols together allow disconnected applications to share identity (users), relationships (memberships, roles, permissions), content (white hot memes) and more.

JLINX actors do not communicate directly. Instead they each emit events on many event streams which one or more other actors listen for. Other actors can react to your event by emitting their own event on a stream that you (or your agent) listen to.

Because micro-ledger streams do not themselves require any token or centralization of any kind, they require no authority to be created and shared and cost only the resources needed to create and share a file.

Some JLINX protocols allow actors to enable other actors to act on their behalf. This is done by generating a cryptographically signed document granting revokable rights to party B to do X, Y or Z on behalf or party A. This mechanism allows JLINX agents to take actions on your behalf.

For non-monetary data exchange (identities, agreements, claims, etc.) multiple Ceramic tiles (tokenless append-only micro-ledgers) interact and are effectively braided together to form eventual consensus.

If you need data tied to a specific blockchain you can think of JLINX as an L2 roll-up and publish batches of references to JLINX ledgers, or specific JLINX ledger entries.

Conclusion

JLINX demonstrates that standard cryptography, JSON documents, human-readable contracts and distributed ledgers based on directed acyclic graph technology are sufficient to build reliable and verifiable claims of relationship, permission and roles, which are independent of transport and do not require consensus on a blockchain.