`SMART CONSENTABLE DATA EXCHANGE WHITEPAPER

Version 8.2: 4th October 2018

JUPITER CHAIN

Creating A New Data Economy With Jupiter Chain

PRELIMINARY DRAFT AS OF 4th October 2018; TO BE SUPERSEDED BY FINAL DRAFT

This is only a draft whitepaper and the information contained herein is subject to further verification, updates, revision, amendments and completion. This draft whitepaper shall not constitute an offer to sell or the solicitation of any offer to buy the tokens described herein nor shall there be any sale of such tokens in any jurisdiction in which such offer, solicitation or sale would be unlawful. This draft whitepaper supersedes in its entirety any other prior marketing materials or other communications concerning any tokens heretofore delivered to prospective purchasers.

Contents

I. Introducing Jupiter Chain

Advanced data analytics such as artificial intelligence and machine learning are advancing by leaps and bounds. In order to fully realise the potential of the technology, access to big data is an important aspect. However, the current way of managing data is outdated and have not caught up with changing regulations and user sentiment. Governments worldwide are putting emphasis on personal data privacy; data owners will control who gets access to their data and how it is being used. The general public is also becoming concerned with how their data is used and if it is being used ethically. Ultimately, entities that seek to collect and monetize data will find it increasingly difficult to be able to do so in the near future if their users does not consent to its usage. Entities holding custodianship of data are going to face higher risks with increased penalties on data misuse and leaks.

Jupiter Chain seeks to pioneer a new paradigm for the data economy by putting the control of data in the hands of the data owners. Entities seeking to assess users through data or learn from big data, will be able to do so without being custodians of the data; consent is retrieved as required. Jupiter Chain is a Smart Consentable Data Exchange built on blockchain technology. The aim is to enable the sharing and analysis of data in a privacy preserving manner; and achieve decentralization in data assessment and analysis.

To this purpose, the Jupiter Chain project aims to practically put together tested blockchain technologies to achieve operationalization without compromising on efficiency and scale. At the same time, it aims to innovate on the treatment of data privacy and security. We start with taking a federated approach with the blockchain network architecture. This takes into account technical, use case and regulatory considerations. In the solution design, we also consider data, identity and consent management. This coupled with a secure smart contract execution environment provides the basis for the smart consentable data exchange.

The envisioned outcome is two-fold. First, to empower data owners with the ability to control and monetize their own data. And second, to enable a decentralized marketplace for data analytics.

II. Platform design - A federated blockchain

The focus of Jupiter Chain is on regional/location specific use cases. Usage of data are mostly within a local ecosystem. There may also be local data residency requirements by regulation. To allow for the flow of digital assets and data across local ecosystems, we implement the Jupiter Chain mainnet to communicate and transfer value across localities. Jupiter Chain is architected to connect local permissioned blockchain ecosystems (known as Local Community Ecosystem Chains, LCEC); allowing for the exchange of data and digital assets across localities. This aims to achieve scalability and efficiency in the following ways:

1. Reduce redundancy on the blockchain by keeping replication of data within its own local ecosystem.
2. Reduce transactional load on the Jupiter Chain mainnet, allowing for the use of public blockchain consensus protocols such as Proof of Stake without compromising on scale.
3. Trust-based consensus used in local permissioned blockchain ecosystems allows for more transactions and throughput to scale within the local network.

It also considers local regulatory requirements. In many countries, financial or even personal data have a local residency requirement; data is not allowed to be stored outside the country or province. This architecture is described in the diagram below:

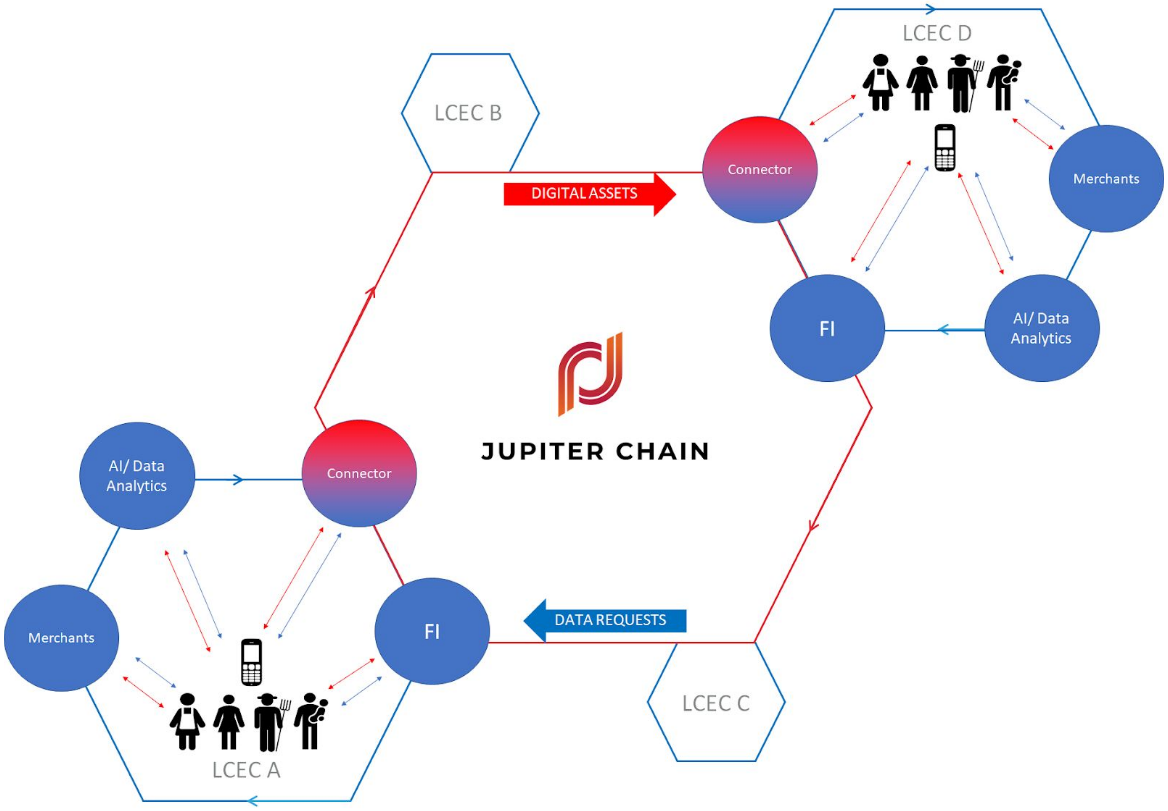


Figure 1: Jupiter Chain platform architecture

Network participants on a LCEC are ecosystem partners of local use cases. They serve as gateways to the data flowing into the network and ensure the integrity of the information. Nodes within the LCEC contribute data on the network via local users, users provide consent for the use of data. This provides the foundation for the exchange of data on the Jupiter Chain platform. In each LCEC, there is at least one node which serves as a connector to the mainnet. Digital assets flow within the LCEC and across to other LCECs via the mainnet. Request to use data for assessment or analytics can be channeled via the mainnet from LCEC to LCEC.

In the development roadmap, LCEC deployment kits will be provided to third parties wishing to set up their own LCECs. This will define the network and hardware requirements as well as provide a guide on how to set up the LCEC framework. We aim to make the onboarding process easy and encourage the organic growth of LCECs. We will explore with cloud service providers to allow for easy set-up of LCEC networks and linking them to the Jupiter chain, leading to one-click services for nodes to onboard onto the network. This will also be coupled with smart contract development kits which will allow for the deployment of the contracts in secure execution environments, this is discussed in the next section.

III. The Smart Consentable Data

With the foundation provided by the blockchain platform, the next consideration is achieving Smart Consentable Data. Data and identity are key to defining consent, this in turn allows for privacy preserving analytics executed in a secure environment.

a. Data, Identity and Consent Management

To set the foundation for data analytics, it is important to define the data structure and allow for flexibility in the data model on the Jupiter Chain platform. In this respect, we implement data handling smart contracts in a modular manner, allowing updates to the data structure without disrupting the execution of business processes.

1. Normalisation in the Jupiter Chain Data Model

Data structures in Ethereum based blockchains tend to be one-dimensional. Relational databases which holds seventy-seven percent of the total market share for data storage uses two-dimensional data structures, that can be easily designed to increase data integrity and remove data redundancy through E.F Codd's normalisation process. We proposed to create data contracts that can define data tables through the use of data layer contracts instead of default data arrays. Consider the following:

An address can contain every data assignable to it as,

$$A = \{D_1, D_2, D_3, \ldots D_n\}$$

therefore, a contract's referenced data D containing a functional dependency where,

$D \subseteq A$, and where $A \in U$

should have a superkey x such that it conforms to,

$$f\text{:}\ \mathrm{x} \rightarrow \mathrm{y}$$

we must consider all transitive dependencies like in,

$R(A,B,C,D)$

to conform to $f{:}x \rightarrow y$ by making all form,

$A \rightarrow B \rightarrow C$, where $A \rightarrow C$

separated into new data sets of,

$A \rightarrow B \text{ and } A \rightarrow C$

and every set of partial and transitive dependencies must be expressed as,

$R_1(X,A_1,A_2 \ldots A_n) \in R$

where X_n is a superkey of relation R_n of R

We separate all dependencies in designing the data model in Ethereum based blockchains to remove any possible data anomalies that might arise from a badly designed data model. Currently, there are no standards for creating the data layer contracts needed in Ethereum based blockchains. Because of this, E.F Codd normalisation process is not facilitated in Ethereum based blockchains. We propose implementing factory method pattern to provide for an upgradable data layer that expresses the above stated characteristics for a well-designed data model.

2. Implementing Factory Pattern for Upgradable Data Layer

A smart contract, once deployed, is considered immutable and permanent in Ethereum based blockchain. The challenge is to allow for easy upgrading of the data layer. Factory pattern has been used to generate objects or data with methods that are easily changed and swappable. We implement a factory-like pattern in the data layer to resolve the issue of smart contract inflexibility. Consider the Factory pattern architectural diagram:

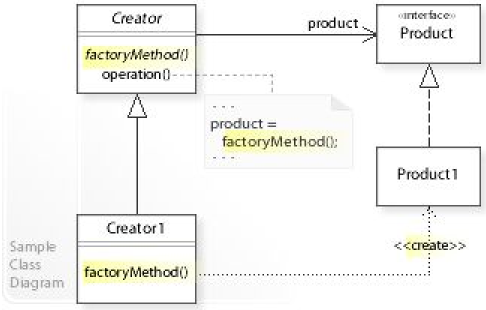


Figure 2: Factory Pattern Architectural Diagram

In the diagram, smart contracts can be assumed to be replaceable with the use of dependency injection through an interface or abstract class. Smart contracts, however; do not support interfaces but can still facilitate swap-ability with the use of an application binary interface (ABI) and smart contract address. We can therefore implement factory-like pattern in Ethereum based blockchain by calling a dependent data smart contract anonymously using its ABI and address replacing the old data contract in the main data contract registry. Please refer to the following adapted diagram:

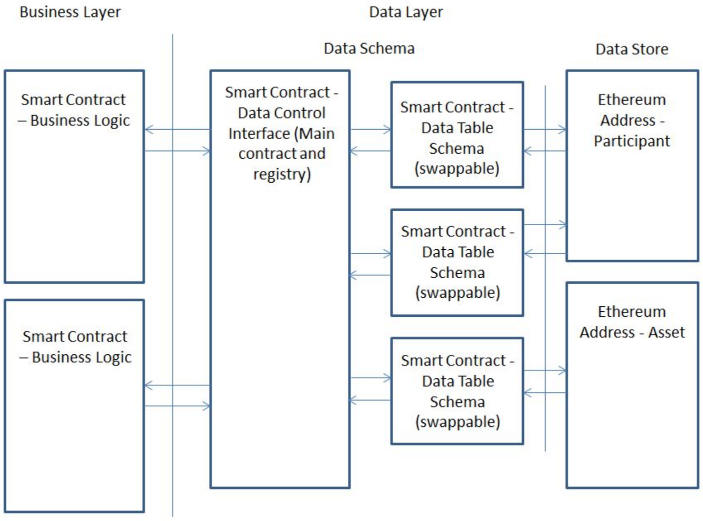

Business Layer
Data Layer
Data Schema
Data Store
Smart Contract – Business Logic
Smart Contract – Business Logic
Smart Contract - Data Control Interface (Main contract and registry)
Smart Contract - Data Table Schema (swappable)
Smart Contract - Data Table Schema (swappable)
Smart Contract - Data Table Schema (swappable)
Ethereum Address - Participant
Ethereum Address - Asset

Figure 3: Smart Contract Data Layer

3. Reducing Computational Complexity of Data Retrieval for Data Science

The data layer proposed should have indexing by addresses and since most operations for analysing the data in Ethereum based blockchains are associated to a particular address, we can expect most data retrieval performance to be of complexity:

$$O(n)$$

This is particularly promising for providing fast customised services to individuals and businesses. Data analytics can be performed near *O(n)* complexity by keeping track of addresses that have consent.

A key element to facilitate the notion of consentable data is data privacy management. On the Jupiter network and LCECs, data will be persistently encrypted and stored in a distributed manner. Data access is managed by a rights management model where the data owners grant consent for access to their data. There are three categories of data on the Jupiter Chain network, transactional, network and identity-related.

4. Transactions within the ecosystem.

This refers to the day to day transactions between users and entities on the network. This could be purchases, sales of goods and services or records of digital assets. It also contains two sub-categories that are of interest:

Transactions with Jupiter tokens. A native token, JUPT, will be issued on the Jupiter Chain mainnet to serve as a medium to provide incentives on the platform for end users and network participants (this is described in section on "Network Economy" below). Such transactions records information relating to account balances and token transfers.

Financing information; financing structures, rates and repayment conduct are recorded on the LCEC. And because these records are unrepudiated and registered on the blockchain, it acts to a certain extent as a deterrent to poor repayment conduct and exorbitant financing rates by irresponsible lenders. Other LCECs can connect with each other and access lending and borrowing opportunities. This also means that peers (and microlenders) across LCECs can also reach each other for remit and financing purposes.

5. Network data

Refers to data on how nodes are connected to each other and how their interactions within the network. Interactions and actions on the network provide good insights into the nature, value and volume of exchanges between peers. In fact, the degree of connectedness amongst peers sitting on an LCEC allows a glimpse into the inter-relatedness of network relationships and potentially to match opportunities of even unrelated peers.

6. Identity-related data

This will be stored on a self-sovereign identity service such as those described by projects such as Sovrin, Civic and Uport. The network can contain a wide range of identity verifiers such as government services, utility providers, internet-based services such as ecommerce and social network sites. The data could consist of identity documents, biometric information, verified mobile number and social connections. This will allow a user to onboard a local community ecosystem chain (LCEC) easily through the self-sovereign identity service. Local services need not manage identity/KYC centrally, this also allows user to port their identity across different use cases. When needed, we can also allow for network validation from linked partners within the local network. Linked partners can be the local village chief, local shop owner, other users, suppliers and buyers.

The diagram below demonstrates the process of onboarding a user onto the blockchain platform. Contact information is submitted by the user's or an ecosystem partner's machine. It can be anything from email address to social media accounts or mobile phone numbers. The message is only stored on the blockchain after verification is done by a server machine.

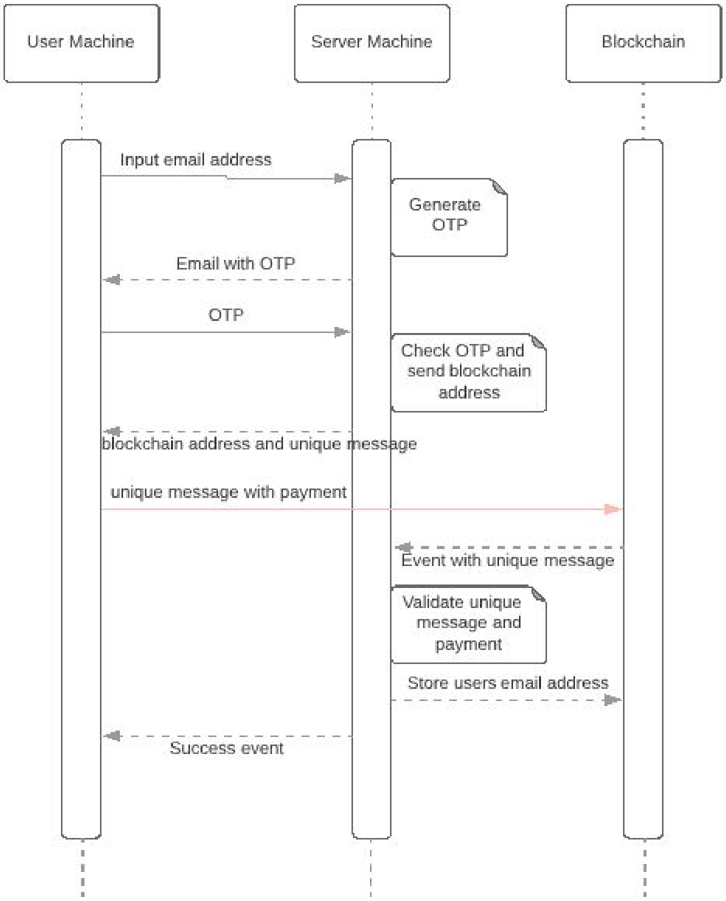

User Machine
Server Machine
Blockchain
Input email address
Generate OTP
Email with OTP
OTP
Check OTP and send blockchain address
blockchain address and unique message
unique message with payment
Event with unique message
Validate unique message and payment
Store users email address
Success event

Figure 4: User identity onboarding flow

In the first stage, contact information is verified as an identity challenge such as a One Time Password (OTP). The next stage is giving the blockchain information, which triggers validation and storing of the actual contact information.

By bringing together identity-related information into one platform, users can form a strong identity profile. To incentivize this in the onboarding process, network tokens can be used as rewards. For example, they can earn points or JUPTs by linking their account to a phone number, upload a picture and have someone endorse the picture. The data points which will be retrieved through the service depends on the onboarding requirements of the use case. Consent to access the service will be provided by the user through a decentralized key management system. Keys are not centrally controlled, allowing for more autonomy. We discuss this in the next section. The diagram below describes the process for identity verification.

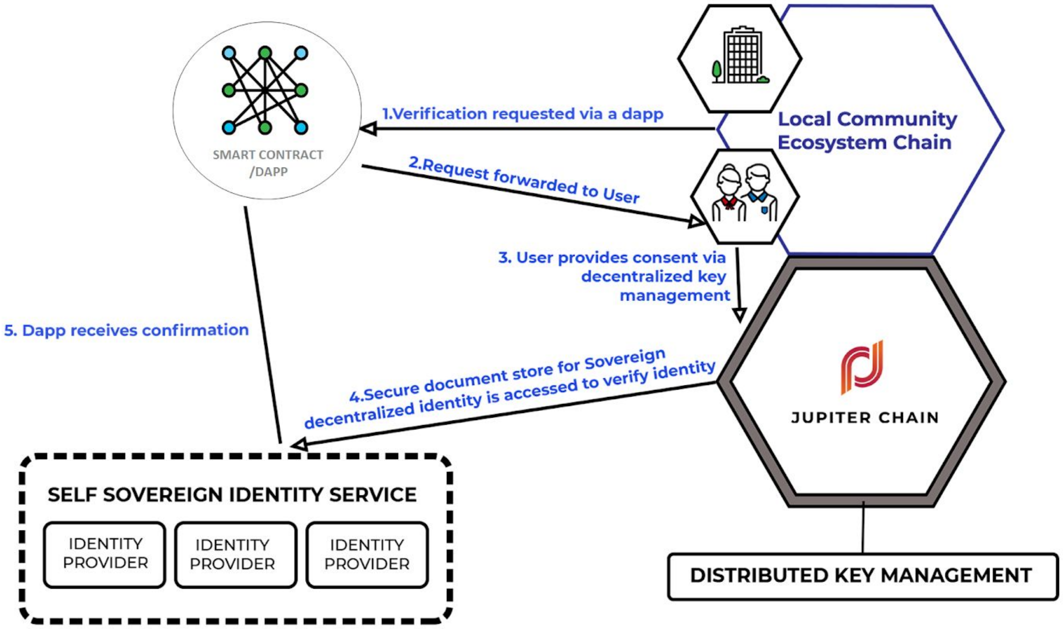

SMART CONTRACT /DAPP
1.Verification requested via a dapp
Local Community Ecosystem Chain
2.Request forwarded to User
3. User provides consent via decentralized key management
5. Dapp receives confirmation
4.Secure document store for Sovereign decentralized identity is accessed to verify identity
JUPITER CHAIN
SELF SOVEREIGN IDENTITY SERVICE
IDENTITY PROVIDER
IDENTITY PROVIDER
IDENTITY PROVIDER
DISTRIBUTED KEY MANAGEMENT

Figure 5: Verification of identity via a self-sovereign identity service

7. Controlling consent with distributed key management

User consent to access data is central to the Jupiter Chain platform. As such, data encryption and key management an important aspect. A distributed key management system integrated in the Jupiter Chain platform will be used for users to provide consent to smart contracts or dapps requesting access to the data. Technical considerations include efficiency and security, we can store and verify keys in a key way. A standalone key server is the most common approach, the server will reside on the same machine as a blockchain client. This allows the key server to interact with the blockchain thru the blockchain client. Alternatively, keys can be managed via smart contracts, this can be done either with off-chain or or-chain verification. Below is a comparison between the three methods:

	Key server	**Smart contract (off-chain verification)**	**Smart contract (on-chain verification)**
Server required	Yes	No	No
Editable by attacker	Always	Before publishing	Never
Data can be taken down	Yes	No	No
Replication	No	Massive	Massive
Costs	Admin	User and admin	User

Data is protected via encryption before it is stored on the blockchain, however it also needs to be protection during execution of analytics. To retain privacy of the data when analytics are run, we plan to utilize secure execution environments in running the smart contracts, this will be covered in the next section. The process of providing consent is described in the diagram below:

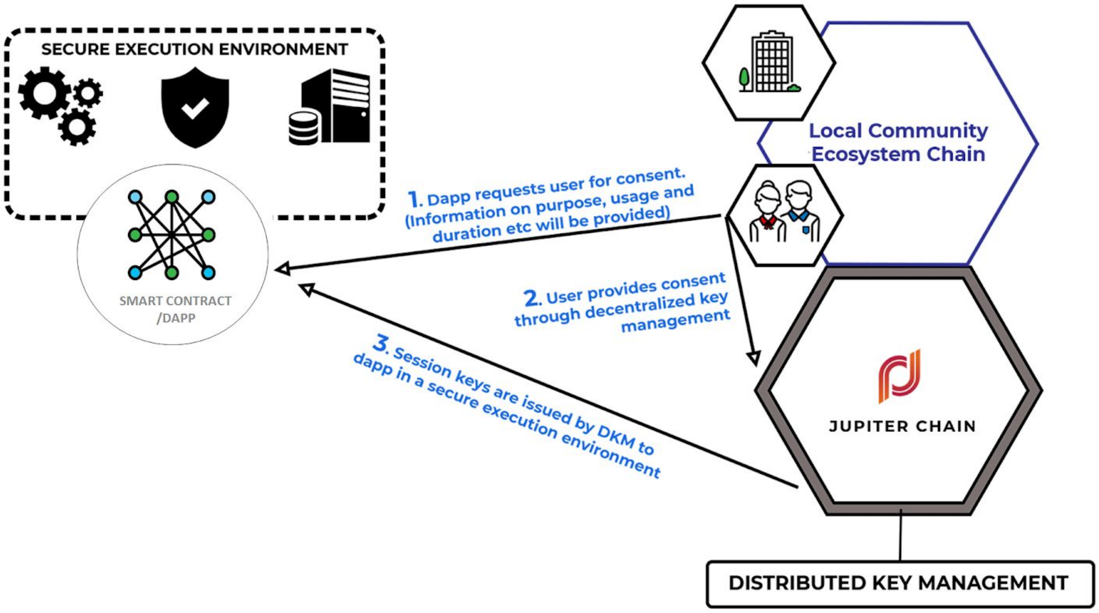

SECURE EXECUTION ENVIRONMENT
SMART CONTRACT /DAPP
Local Community Ecosystem Chain
1. Dapp requests user for consent. (Information on purpose, usage and duration etc will be provided)
2. User provides consent through decentralized key management
3. Session keys are issued by DKM to dapp in a secure execution environment
JUPITER CHAIN
DISTRIBUTED KEY MANAGEMENT

Figure 6: Providing consent via a distributed key management system

When the user (who can be a person or smart contract) grants consent through the key management system, this initiates a key search. This can be initiated via a simple web or mobile application.

The app will basically search the smart contract thru the blockchain client. The diagram below shows how the search starts with the user and ends with the blockchain client. The server application needs to store the address of the smart contract which stores the information needed. This is done thru the blockchain client which resides on the same physical machine.

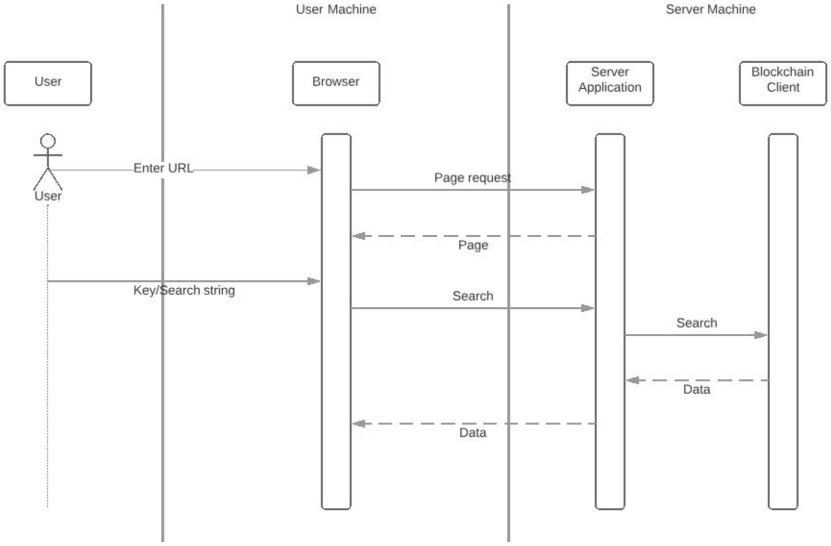


Figure 7: Key retrieval flow

Another consideration is key revocation, this is required when the user no longer wishes to share information with the associated party. Only the data owner will be able to revoke the public key. The first stage for is to verify the owner of the public key. An identity challenge similar to when the user is onboarded will be deployed. Once verified, the blockchain address and unique message will be sent by the server machine. This will be relayed by the client to the blockchain which triggers an event, wherein the server will verify the unique message and once verified will give the revoke command to the blockchain.

b. Secure Smart Contract Execution Environment

The diagram below describes the process where the smart contract executes analytics within a secure execution environment. Analytics are executed within an enclave where data is decrypted after consent is granted.

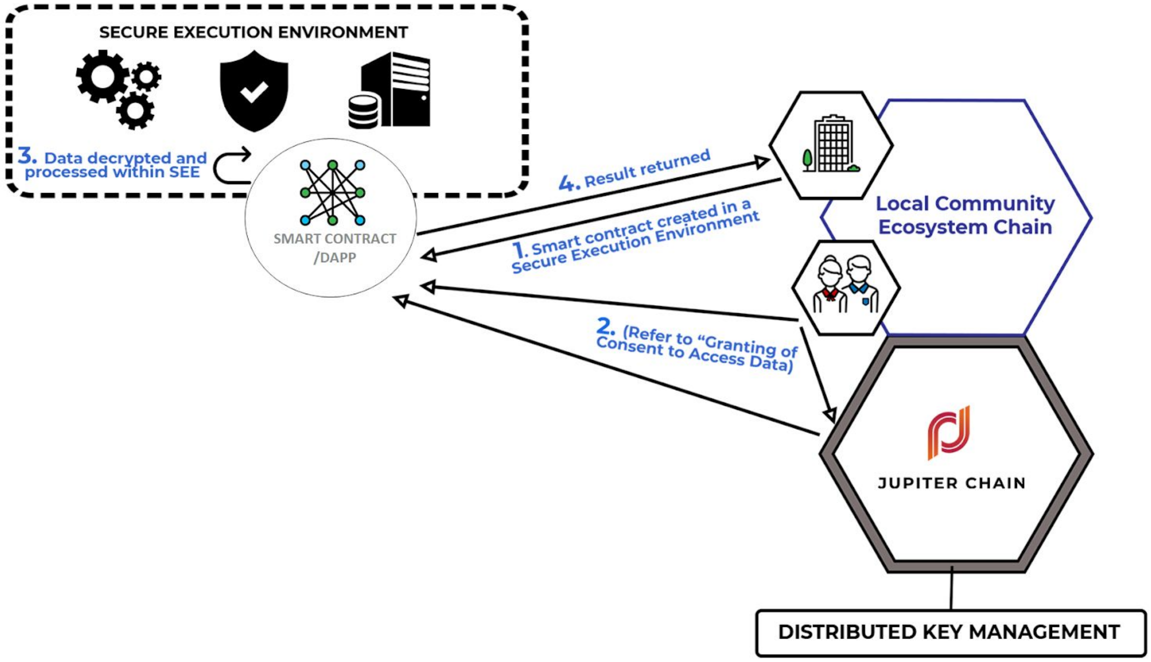

SECURE EXECUTION ENVIRONMENT
3. Data decrypted and processed within SEE
SMART CONTRACT /DAPP
4. Result returned
1. Smart contract created in a Secure Execution Environment
2. (Refer to "Granting of Consent to Access Data)
Local Community Ecosystem Chain
JUPITER CHAIN
DISTRIBUTED KEY MANAGEMENT

Figure 8: Privacy preserving analytics within a secure execution environment

Figure 9 provides an overview of how the Secure Smart Contract Execution Environment works. Client inputs (1) and smart contract state (2) which are encrypted outside of secure enclaves are retrieved, and are then decrypted within secure enclaves (3) where computation and smart contract execution occurs (4).

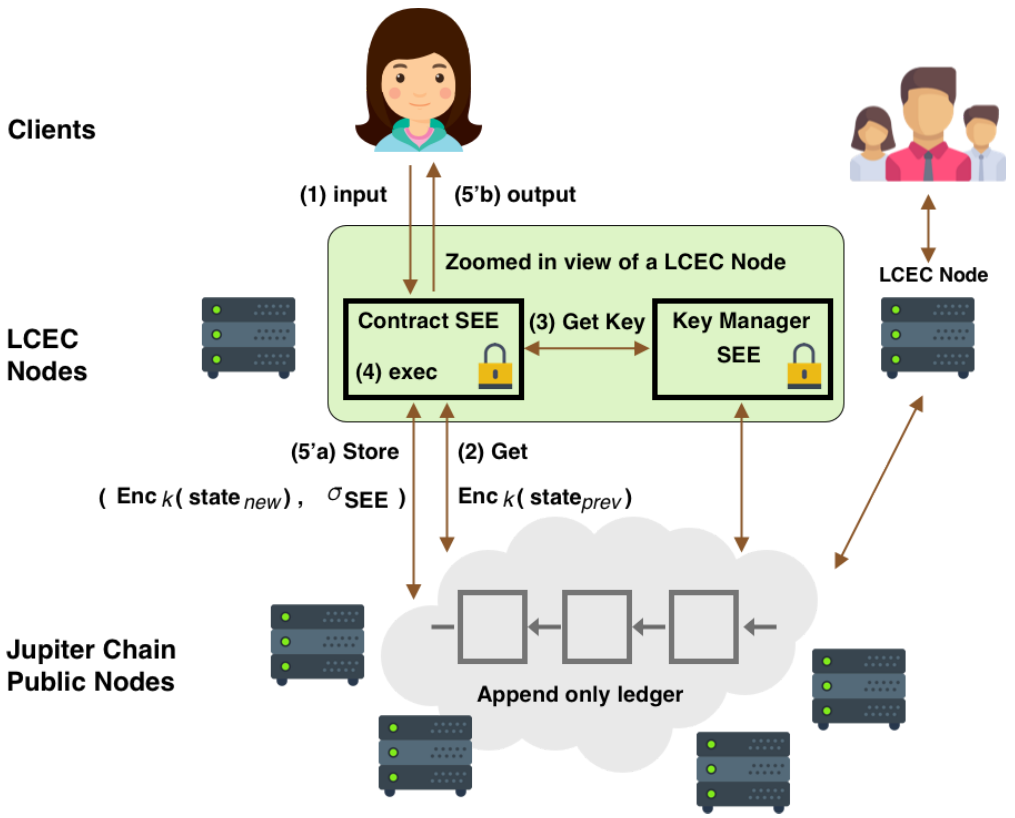
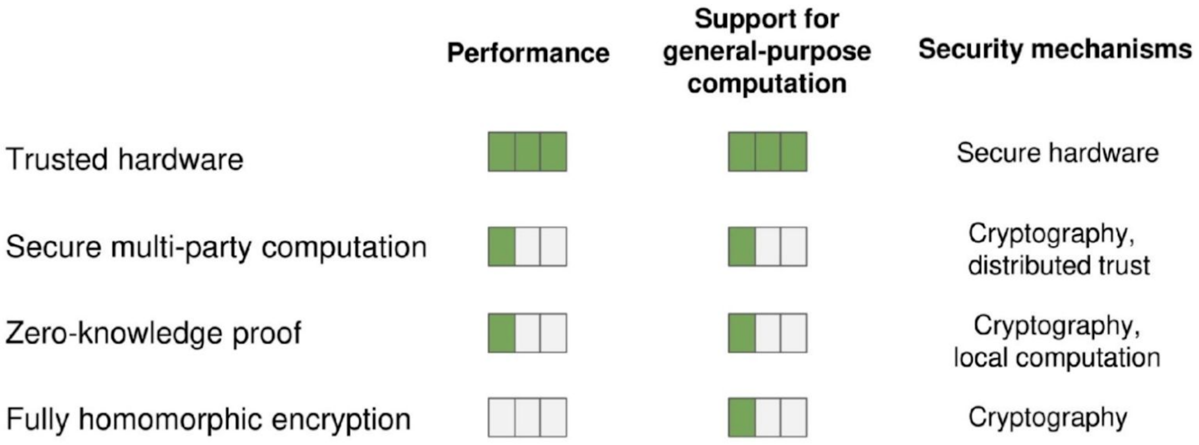


Figure 9: Secure Smart Contract Execution Environment

The resulting smart contract state and results are re-encrypted again before being stored back on the network. After results are computed, other secure enclaves in the network attest the integrity of the results. We make heavy use of Trusted Execution Environments (such as Intel SGX) to execute smart contracts off-chain. The technology is widely accessible for consumers (more accessible than say, setting up a mining rig).

Privacy on the blockchain often involves large trade-offs with performance. Techniques such as ZK-SNARKs and secure multi-party computation (SMC) produce large performance overheads for simple shielded transactions on the blockchain. An overview of competing privacy-preserving approaches and their deficiencies are outlined in ***Figure 10***. Most are unable to fully support general purpose computation (addition, multiplication, modulus, etc.).



	Performance	Support for general-purpose computation	Security mechanisms
Trusted hardware	🟩🟩🟩	🟩🟩🟩	Secure hardware
Secure multi-party computation	🟩⬜⬜	🟩⬜⬜	Cryptography, distributed trust
Zero-knowledge proof	🟩⬜⬜	🟩⬜⬜	Cryptography, local computation
Fully homomorphic encryption	⬜⬜⬜	🟩⬜⬜	Cryptography

Figure 10: Comparing Privacy Preserving approaches

The secure smart contract execution environment allows for two key use cases for privacy preserving analytics. First, it enables smart contracts for data assessment. This involves entities directly assessing a user's data for purposes such as the provision of financial services. The assessor will deploy a decentralized application which contains the assessment criteria. The decentralized application will be given rights to access and process the data; the assessor will only see the results of the assessment. This provides enhanced privacy over the individual's personal information.

Second, it allows for smart contracts as data subscription services, this refers to collection of data from target groups for the purpose of data analytics such as machine learning. In this case, the data requests are propagated to peers on the Jupiter Chain and relayed to individuals via a decentralized application. The decentralized application will then aggregate the data in an anonymous manner. The data can be indexed if required for longitudinal studies.

The framework for these two types of contracts will be developed into SDKs and made available to third party service providers wishing to provide data assessment services or conduct their own analytics on the Jupiter chain platform.

IV. Network economy: Tokens and Consensus

Jupiter tokens (JUPT) is the native token on Jupiter chain. They will be initially issued as ERC20 tokens. At a later stage when the Jupiter chain is deployed, the ERC20 tokens will be swapped into the native tokens. The initial swap will be for pre-mined tokens corresponding to the number of tokens issued during an earlier distribution if there are any. JUPT serves several utilities on the platform, we outline its purpose for four types of stakeholders:

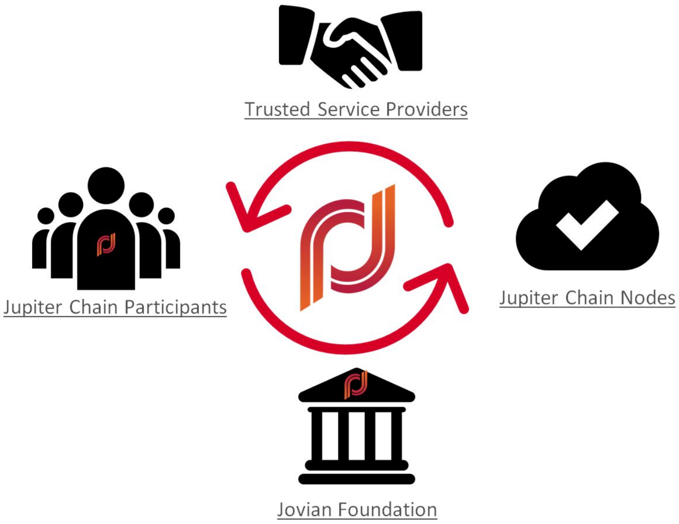


Figure 11: Stakeholders on the Jupiter Chain network

Stakeholder 1: Jupiter Chain Participants

General users of the Jupiter chain earn JUPT by contributing and consenting access to their data. They can then use JUPT to pay/offset purchases and transaction fees from ecosystem partners. In the long run, individual JUPT token holders can also lend and transfer JUPT tokens and be part of a network peer to earn rewards for ‘inclusion’ and supply chain actions conducted on the network.

Stakeholder 2: Trusted Service Providers

Service providers and data companies can utilize the Jupiter Chain to understand the risk-benefit considerations of serving different LCECs and open a whole new scale of market opportunities. These companies can request access to data and pay JUPT directly to incentivize and reward data owner. To do this, they can purchase analytics from other service providers on the Jupiter Chain with JUPT or provide their own. Similarly, third party applications can be offered on the platform by such providers and JUPT used to pay for these services.

Stakeholder 3: Jovian Foundation

Jupiter Chain's evolution will largely be community driven and the platform will ultimately be open-sourced. As such, the Jovian foundation is set up with the mandate to lead the stewardship of digital assets and funds raised throughout the life of the foundation and going concern of Jupiter Chain. To ensure sustainability of the foundation, JUPT will be collected as fees for various transactions executed on the chain. Service provider nodes also pay network fees to access the network. Jovian foundation's role covers the following activities:

a. Governance and compliance

Appointed directors and members of the foundation will ensure the setting up of adequate governance frameworks that comply to local laws and jurisdictions that the foundation operates in. The aim is to ensure accountability and proper stewardship of all activities run under the foundation.

b. Technology research and development

The foundation or its appointed vendor(s) are responsible for the construction of the Jupiter platform and ongoing research and development that covers upgrades, maintenance, licensing, SDKs and other technical services.

c. Ecosystem building and adoption

An important role of the foundation is to build and manage Jupiter's community and ecosystem efforts by encouraging adoption and providing support that helps scale Jupiter network. This includes business development activities, expanding ecosystem partnerships and providing required technological support and solutions.

Stakeholder 4: Jupiter Chain Nodes

The Jupiter chain will be a public blockchain where JUPT circulates to the local community ecosystem chains; it will also relay data requests across the local community chains. These interactions will be validated by peers on the blockchain through its consensus protocol. Nodes can also earn JUPT for validating transactions on Jupiter Chain.

a. Mainnet to LCEC: Collateralised Staking

Since an LCEC only consists of trusted nodes which are part of the local community, it can function on RAFT or BFT (Byzantine Fault Tolerance) which are commonly used in permissioned blockchains. This allows for faster throughput on the network. These trusted nodes will also participate on the Jupiter chain mainnet to allow for the flow of tokens, digital assets and data requests into the LCEC. They also benefit through collateralized staking.

In order for JUPT to flow into a LCEC, a trusted node needs to provide a custodian service for JUPT on the mainnet. Trusted ecosystem partners provision nodes on the LCEC and channel JUPT into the LCEC to facilitate rewards and transactions. By doing so, the node can participate in transaction validation on the mainnet via proof of stake or collateralized staking. This is described in the Figure 12.

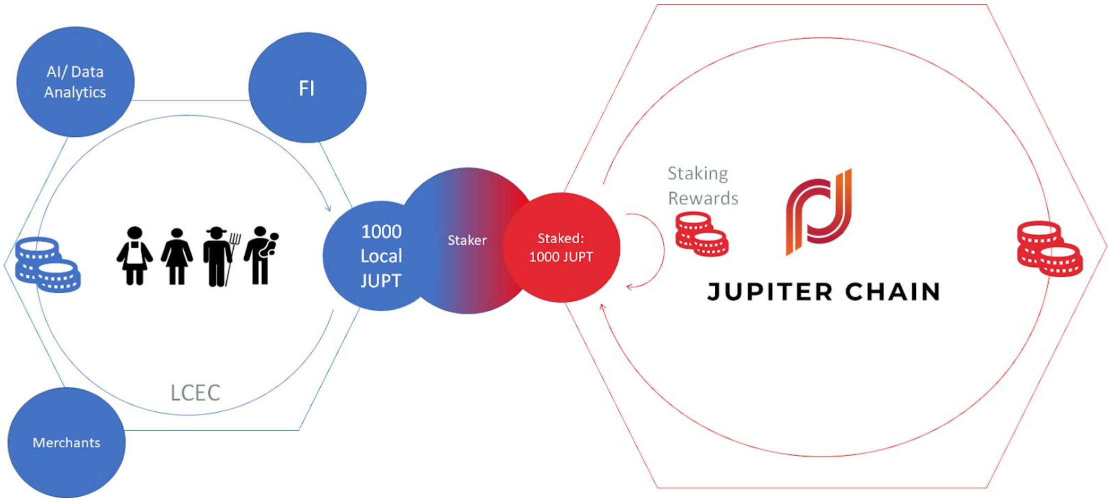


Figure 12: Collateralised Staking in the Jupiter Chain

This provides the node with the equivalent of custodian fees and incentivizes them to provide this service. The minimum collateralized amount for each node can be determined by the size of the LCEC (number of users and number of nodes). The Jupiter chain is also open to non-LCEC nodes who wishes to participate and provide validation on the network. These nodes will not be subject to the minimum stake requirement.

b. Jupiter Protocol: Time Framed Proof of Stake

For the Jupiter chain platform to sustain in the long run, there is a need to ensure continued incentive to participate in validation on the network and token price stability. Hoarding of tokens is a common strategy taken by token holders in a public blockchain network, this makes the supply of coins in the market inconsistent and causes much volatility in the market. One approach to counter such a strategy is to incentivize coin owners such that they are encouraged to circulate coins in the network. We propose a time-framed approach for proof of stake to support this. Each trusted node can appoint a selected public address as its staking address. The node will be allowed to participate in validating transactions as long as there are stake-able coins in the staking address. A coin is stake-able[1] if it fulfils the following conditions:

1. Maturity. The coin must exist in the staking address for M blocks.
2. Recency. The coin should not have existed in the staking address for R blocks.

A node will choose to stake coins as long as the expected rewards are larger than the best alternative. The alternative in this case could be selling the coins in the market.

Let’s first start with the simple case of the staker owning only one coin. Assume that B is the block reward. The probability of winning the block is a function of the total number of coins

[1] Using this method, one can differentiate between stake-able and non stake-able tokens within the staking address. This can be achieved without the need for non-fungible tokens. Non-fungible tokens for this purpose would impose unnecessary costs on the platform to account for and issue.

staked (S) on the network. The expected reward per coin staked per block is thus= (1/S)(B) = (B/S). Assume that stakers are risk averse, this expected value becomes (B/S)^ai, where ai is staker i's level of risk averseness, this ranges from 0 to 1 and is capped at 1 where the staker is risk neutral. The maximum number of blocks a coin can be staked for is given by R-M. The expected reward in fiat currency for staking a coin to staker i for the maximum period of R-M is given by:

$$(\frac{B}{S})^{ai}\,[\frac{1-(1+r)^{-(R-M)}}{r}]\,F(t=t+R-M) \qquad [JP.1]$$

Where, F(t=t+R-M) is the expected exchange rate at block period t+R-M. If instead the staker sold the coin, his gets F(t) or the value at the coin at the current exchange rate. F(t) the token price at time t is a function of the total amount of coins being put up for sale as well as the network demand for the token.

Now, let's assume that the staker is endowed with X number of coins. The staker has to decide how much of his coins to stake and how much to sell. As he increases the number of coins staked, the chance of winning becomes higher. His expected reward for staking Y number of coins for R-M periods is:

$$(\frac{YB}{S})^{ai}\,[\frac{1-(1+r)^{-(R-M)}}{r}]\,F(t=t+R-M) \qquad [JP.2]$$

If he sells the rest of his coins he will get (X-Y)F(t). In equilibrium, the amount of coins staked is the Y that solves $(\frac{YB}{S})^{ai}\,[\frac{1-(1+r)^{-(R-M)}}{r}]\,F(t=t+R-M) = (X-Y)F(t)$. The number of coins staked and traded at any given period depends on the market dynamics.

i. Optimal strategy - token cycling

If demands grows at a steady rate, the network should reach an equilibrium where each node stakes a stable amount that is based on the staker's risk averseness and endowment. In such a case, the staker should adopt an optimal strategy of token cycling such that the number of stake-able coins are fixed. Take for example, if its optimal for the staker to stake 1000 JUPT at any given period. Since coins expire from the stake-able pool, it is optimal to sequence the expiry such that a fraction of the coins expire at a regular interval. The staker sells the expired tokens in the market and acquires new tokens to replace them. At any point, the staker needs to hold M number of additional fractions of the tokens such that they are maturing in the staking address. This strategy also needs to consider the potential token rewards that are earn from staking.

This will ensure that there is a constant stream of tokens being supplied by stakers in the market and that tokens are being circulated. The above is a proposition for the design of a time framed proof of stake protocol. Research and development is currently being tested with agent-based computer simulations with the aim to progress testing to a testnet environment. If feasible, we believe the protocol will provide adequate incentives for network participation in the long run as well as encourage stability in the token market.

V. Applications of Jupiter Chain

Jupiter Chain is about socio-economic inclusion and enabling everyone to be part of the data economy. This can be applied to many areas. In finance, wealth management and investment products can be tailored precisely to suit an individual, considering her past, current and even future potential. Health and social assessments based on psychometric and lifestyle data serves important research and insights for new products and services that strive to promote better quality of life. Innovative and nimble technology solutions especially those with an inclusion and 'tech for good' agenda can thrive against larger incumbents and bring true value to consumers.

For all sorts of uses cases, once sufficient data has been accumulated on the platform, entities looking to analyse big data can send out smart contracts with their data requirements. This information gets propagated to the relevant data owners and consent is obtained. Data owners will be reward directly for providing their data. This data is then anonymized and aggregated by the smart contract. This sort of request can be either one-off or for continuous feed. For continuous feed, the smart contract will index the data to enable analysis on panel data. Through such big data requests, algorithm providers in the first case can tap onto the ecosystem and use machine learning to refine their assessment modules. In this section, we explore key use cases where Jupiter Chain can be applied. This list is by no means exhaustive.

a. Financial Services

Financial services are a natural fit for the Jupiter chain platform. Stricter personal data privacy and know your customer regulations are increasing compliance costs. Custodianship of client's data is becoming a burden in terms of compliance risks. Jupiter chain will not only allow users to share data across financial institutions but also manages user data and user consent to deliver privacy preserving analytics. Taking away the associated risks with holding user data and yet providing trust for data integrity.

Financial institutions (FI) will be trusted ecosystem partners by running nodes on a LCEC. Using Jupiter Chain SDK's, these FI will be able to deploy smart contracts which will perform assessment on-chain with either their own algorithms or tap onto third party assessment services which are provided on the Jupiter Chain. For example, they can identify individuals that best fit certain products offerings in the platform (with the user's consent). Or when a user requests for financial services such as a loan on the Jupiter Chain platform, she can provide direct permission to utilize the data for assessment.

b. Health and lifestyle services

Medical and health related data are private to the owner. Jupiter Chain can provide the platform to securely store such personal data and yet allow sharing of medical histories when a patient goes to a new doctor. Trusted medical services and healthcare providers will serve as nodes on such a platform.

With this data, users can also get access to tailored services such as lifestyle services or insurance. Such data can be used to develop customized products such as cosmetics, meals and fitness plans. The more information the user gathers on the platform the better companies will be able to customize products to users. Jupiter Chain will allow such sharing without exposing the raw data.

c. Retail rewards and loyalty points

Rewards and loyalty points are issued by retailers to encourage return customers and loyalty. In the issuance and utilization of loyalty points, more information such as shopping and dining habits are collected. Such data can be analyzed by retainers to roll out campaign, promotions and even new products and services. It can also be used by financial institutions to provide target services like credit cards and loans. Jupiter chain can provide a platform for the storage and sharing of such data for the purpose of retail market research. Consumers can have the peace of mind that their personal data will not be exposed to retailers without their consent.

VI. Research, development and deployment plan

Research, development and implementation of the Jupiter Chain vision will take place in 4 tracks. Figure 13 breaks this down in the timeline.

Track 1. Data

This track is focused on identity, data structure and the handling of encrypted data on the blockchain. The identity solution to be implemented on the platform will be based on existing implementations for self-sovereign identity and tailored for our LCEC use cases. Data structure will be implemented to be modular and flexible, this allows for easy onboarding of LCECs. Handling of data is an important consideration for the consent layer, where and how data is decrypted matters in keeping the data private and the system efficient. This provides the foundation for all data on both the mainnet and LCECs.

Track 2. Consent

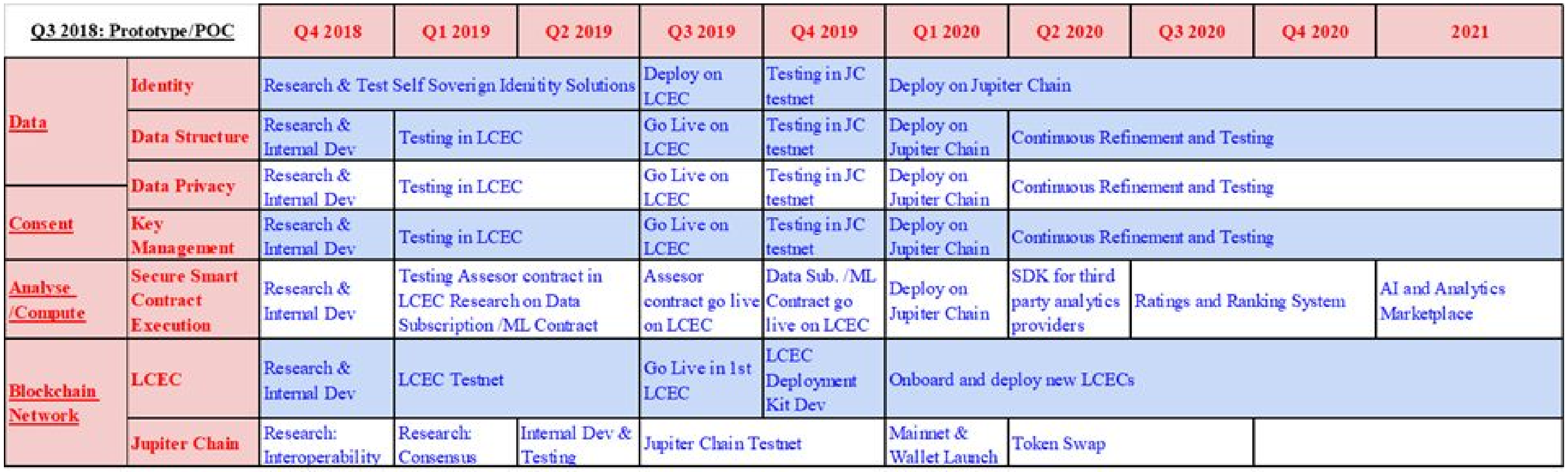
This track is focused on developing technology to enable the control and permissioning of user data. Encryption of data and how private keys are managed and maintained on the platform needs to be considered in this track. The plan is to test and deploy first on an LCEC testnet before it goes on the mainnet.

Track 3. Analyse / Compute

This track focuses on enabling the computation of smart contracts for analytics in a secure execution environment. Research will focus on looking at current developments in the area and innovating on applications to Jupiter chain use cases. This track will start with testing for the data assessor smart contracts and move on to smart contracts as a data subscription service. The aim is to enable machine learning to be done within an enclave keeping the data away from the deployer of the analytics. Once this is tested and ready, SDKs will be developed and made available for third party developers to build decentralized applications that will conduct analytics on-chain. Finally to enable the AI marketplace, a ratings and ranking system will be built for users to compare analytics services.

Track 4. The Blockchain network

This track includes deploying technology on the ground, testing and development of local community ecosystem chains (LCECs) and the Jupiter chain mainnet. This track considers interoperability between the mainnet and the LCECs as well as the Jupiter consensus protocol.



Q3 2018: Prototype/POC		Q4 2018	Q1 2019	Q2 2019	Q3 2019	Q4 2019	Q1 2020	Q2 2020	Q3 2020	Q4 2020	2021
Data	Identity	Research & Test Self Soverign Idenitity Solutions			Deploy on LCEC	Testing in JC testnet	Deploy on Jupiter Chain				
	Data Structure	Research & Internal Dev	Testing in LCEC		Go Live on LCEC	Testing in JC testnet	Deploy on Jupiter Chain	Continuous Refinement and Testing			
	Data Privacy	Research & Internal Dev	Testing in LCEC		Go Live on LCEC	Testing in JC testnet	Deploy on Jupiter Chain	Continuous Refinement and Testing			
Consent	Key Management	Research & Internal Dev	Testing in LCEC		Go Live on LCEC	Testing in JC testnet	Deploy on Jupiter Chain	Continuous Refinement and Testing			
Analyse /Compute	Secure Smart Contract Execution	Research & Internal Dev	Testing Assesor contract in LCEC Research on Data Subscription /ML Contract		Assesor contract go live on LCEC	Data Sub. /ML Contract go live on LCEC	Deploy on Jupiter Chain	SDK for third party analytics providers	Ratings and Ranking System		AI and Analytics Marketplace
Blockchain Network	LCEC	Research & Internal Dev	LCEC Testnet		Go Live in 1st LCEC	LCEC Deployment Kit Dev	Onboard and deploy new LCECs				
	Jupiter Chain	Research: Interoperability	Research: Consensus	Internal Dev & Testing	Jupiter Chain Testnet		Mainnet & Wallet Launch	Token Swap			

Figure 13: Research, development and deployment timeline

VII. Conclusion

Jupiter chain aims to enable a new paradigm for the data economy where data can be shared with confidence that privacy can be maintained and that data owners can be directly rewarded for their contribution. Jupiter chain aims to realise capabilities to collect, share and analyze massive smart consentable data; by deploying a distributed network on blockchain with a tokenized incentive structure to distributing gains from data owners to data users.

As third-party service providers onboard to Jupiter chain, a marketplace for AI and machine learning algorithms can emerge where all users on the platform (no matter large or small) can tap on to these expert services. Small technology companies can get access to data that would be crucial for their product development. Large companies can leverage on data points outside of their institution, providing a more well-rounded view of the user.

Products and services such as savings, investments, insurance and e-commerce can be tailored to each consumer's unique characteristics. Opportunities to develop new products and services will surface; creating a win-win for all parties. This creates a truly inclusive data economy with a level playing field for all.