Smart Layer: A Decentralised Integration Protocol for the Next-Generation Web

1st September, 2023

Abstract

In the ever-evolving decentralised Web, a robust integration layer bridging siloed services through tokenisation is crucial. Smart Layer is a decentralised network designed to facilitate the next generation of web use-cases. As a network, it seeks to address the inherent limitations of the current Web. This whitepaper delves into the architecture, design, and potential of Smart Layer, and its token.

Introduction

The Web has grown increasingly integrated with the advent of Web 2.0. Integration such as, "Login with Google," "Checkout with Apple," and "Share with Twitter" are hallmarks of the Web2 user experience. The success of tech giants like Google has underscored the demand for integrated services. However, despite these advancements, the broader web landscape has remained fragmented. Centralised entities have emerged as dominant forces, creating isolated ecosystems that limit true integration. Though blockchain technology introduced a new paradigm which emphasised decentralisation and trustless transactions, its primary focus has been on asset tokenisation.

Smart Layer is a decentralised protocol that aims to reshape the Web's architecture. It acts as an integration hub which enables smooth interactions between various services, much like how many websites use platforms such as Google. Smart Layer goes beyond acting as a "bridge," and paved the way for the concept of Smart Tokens. Smart Tokens are tokenised digital rights and products/services that can be seamlessly integrated across various web use-cases. In that sense, Smart Tokens can surpass the limitations of centralised systems, and leverage the strengths of blockchain. This protocol is designed to function as a distributed network, serving as the backbone for the next generation Web.

TokenScript—an evolving OASIS standard—is critical for Smart Tokens within the Smart Layer. While Smart Layer offers the foundation, TokenScript defines the packaging, distribution, and operation of these tokens. TokenScript also establishes the messaging format between the tokens and their integrations. This ensures that they work within the defined parameters of trust, interoperability, privacy and security.

Scope of this document

This whitepaper serves as an introduction to the novel concept, models and mechanism of Smart Layer. It discusses its potential applications in the next-generation Web. It is not intended as a guide for the implementors of Smart Layer Nodes (which is addressed in a future separate specification, akin to Ethereum's Yellow Paper).

Problem statement

Dr. Gavin Wood has attributed the centralization of the web to a combination of factors. These factors include network effects, economies of scale, big data ownership, and intellectual property laws. The centralization Dr. Wood identified has led to a situation where most websites today are integrated with integrations like Google or Facebook login, or Apple Pay, with several websites using all those integrations listed.

The harm caused by web centralization notwithstanding, it has enhanced usability tremendously. But it has also stalled the development of new integrations. This is especially evident in areas where world-renowned internet companies have no major influence, such as with airlines and purchasing tickets for flights. Central points in those areas (see Amadeus in flight booking as an example), can reach only the back offices of various websites. As a result, a flight ticket booked on one website can also be used to book a car rental on the same website, but that website only. Similar to the airline industry, in social media or e-commerce, a user reputation would work on one website exclusively.

This lack of user-level integration can be attributed to two (2) main factors.

First, despite the aforesaid explanations, internet centres have become trust anchors. Should an uncommon third party provide integrations (consider the prospect of using or being provided a calendar other than Google Calendar), Whether a user's trust will be "betrayed" is brought into question. Before blockchain emerged in the form of a trust-machine that could facilitate trustless interactions between parties, this trust problem was not resolvable without massive, centralised Internet companies as the trust anchors.

Second, the complexity of the system grows quadratically with the growth of the number of integrations used.

Having regard to these two (2) reasons, the modern Web faces a "Limit of 3" challenge. Most websites are restricted to three (3) main integrations: login, social media posting, and checkout.

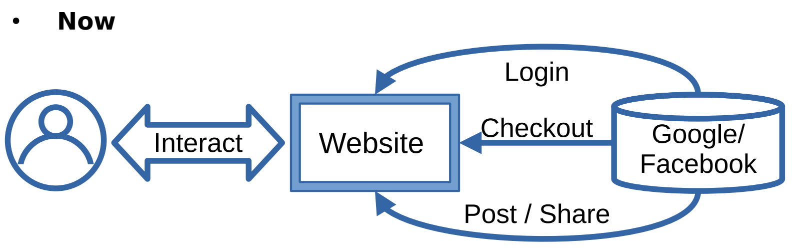

Figure 1: Present Web relies on central integration points. They are the trust anchor of the Web

The Web's fragmented nature has, in turn, led to fragmented user experiences. Reconsider the example of an airline flight ticket. In the current web paradigm, this ticket represents a token of value within its issuing platform, but it remains isolated. The potential for the ticket to integrate with other systems—updating travel statuses on social media, guiding users via mapping services, or communicating flight changes to hotel booking systems—remains largely untapped. Such straightforward integrations, though long overdue, are hindered by the Web's compartmentalized structure, where centralized entities offer piecemeal solutions.

Web fragmentation highlights the need for a paradigm shift towards a more dynamic and interconnected web ecosystem. This shift would consciously break the "Limit of 3," thereby allowing websites to connect to a bigger ecosystem outside the control of the current Internet centres, while facilitating an integrated user experience. Naturally, such a new paradigm must include a freely grown integration network, low integration costs, and designs for secure, privacy-preserving mechanisms to expedite expansive integration.

Proposed solution: the Smart Layer approach

To reignite web innovation and overcome centralisation issues, we must look beyond merely creating isolated systems that sidestep the primary web integrations of the present. In their stead, we propose to build an integrated web where tokens are the main integration points. By transforming these tokens into web services, we pave the way for the next-generation Web.

Before the advent of the blockchain, creating such an integration system was impossible. Any entity operating it would inevitably become a new central trust anchor. However, the emergence of public blockchains like Ethereum has changed things. Blockchains introduced smart contracts that can be executed securely, offering a trust foundation that does not rely on the goodwill of centralised parties.

But just executing smart contracts securely is insufficient for the integration demands of the next-generation Web. Though smart contracts can define and enforce rules, they do not actively perform tasks. They will not notify a user's mobile phone about a delayed flight, or interface with a healthcare system to offer a diagnosis, even if the flight tickets and user's health profiles are tokenised. These functionalities are expected from a highly integrated web that offers a seamless user experience. To bridge this gap, we need a network providing services built atop smart contracts, playing the role of integration providers not unlike Google's login and Apple's payment integration. The Smart Layer network would fulfil that purpose.

Smart Tokens: The Heart of Integration

Smart Tokens are tokenised representations of digital rights and services at the heart of Smart Layer's design. These tokens are based on blockchain smart contracts and enable limitless integration across diverse web scenarios, challenging the constraints of traditional centralised systems.

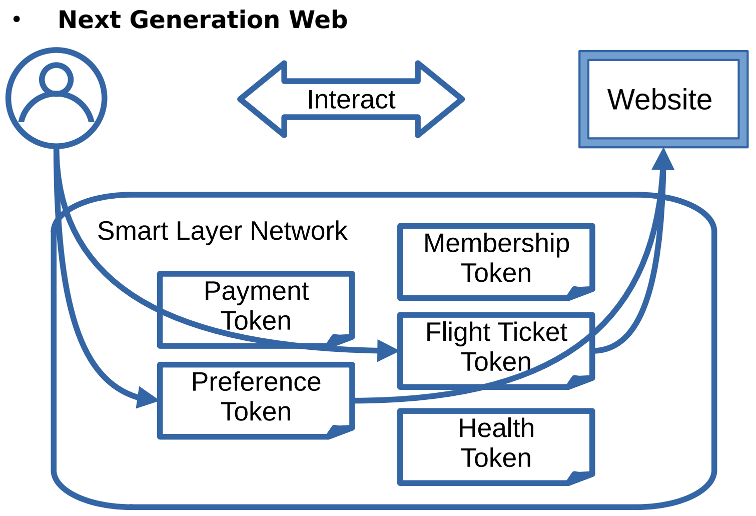

Figure 2: In the next web, users choose smart tokens to use on websites, with unlimited potentials

While directly replacing established systems like Google Login or Apple Pay with their smart token counterparts might seem impractical given the ubiquity of these services, the true value of Smart Tokens lies in their potential for innovation. The diagram showcases the myriad of smart tokens that can be crafted using the synergy of TokenScript and the Smart Layer network. Instead of a blanket data reveal, users can selectively deploy these tokens, maintaining control over their digital interactions. Notably, unlike dApps, these tokens are not designed to replace Web2 but to coexist and enhance the web2 ecosystem. Traditional services like Google Login or Apple Pay remain accessible, but innovations like the smart flight ticket carve out new avenues for web evolution.

With Smart Layer, users navigate with an arsenal of smart tokens, each tailored for specific applications, thereby unlocking a realm of untapped potential. The end goal is a web that's dynamic, interconnected, and pivots around user agency.

Protocol Requirements

Smart Layer's design and functionality hinge on protocol requirements crafted for the distinct capabilities of smart tokens. These requirements are not merely a reflection of standard practices for distributed networks but are intricately linked to the challenges and goals of the Smart Layer ecosystem. The key areas of focus include:

- **Authenticity**: The integrations should be able to verify the authenticity of the result of the Service TokenScript code executed on the smart layer network, and the network shouldn't rely on the integrations verifying this alone for operational integrity.
- **Serviceability**: This encompasses continuous uptime, redundancy, and load balancing. While mature industrial technology can meet these requirements, their application within Smart Layer is influenced by other interconnected requirements.
- **Privacy and Security**: Smart tokens distribute their logic between user agents (like decryption of sensitive data) and server-side logic (such as triggers set within the tokens). This paper primarily addresses the server-side logic executed by the Smart Layer network.
- **Token Lifecycle Management**: This pertains to the management of smart tokens throughout their existence. Considerations include the duration a flight ticket smart token resides on a node and the mechanisms to reinstate tokens that are in dormant states, such as a car-insurance token awaiting activation.
- **Inter-node Collaboration**: Nodes within the Smart Layer network are expected to work together to facilitate specific token functions. For example, a smart car token's status could be influenced by its registration, insurance, and maintenance tokens, potentially managed on different nodes. Integrations expect smooth interactions between nodes, allowing them to concentrate on the capabilities provided by smart tokens rather than managing their intricacies.
- **Incentive Structure**: The cost of operating a token is typically borne by the integration. For instance, if a health token is used by a website to optimise a user's shopping list, the e-commerce platform incurs the cost. However, token issuers play a pivotal role in ensuring the availability and operability of their tokens on the network. They must incentivise the network to maintain the token's availability, even as the actual operation costs are met by integration points.

It is essential to differentiate these requirements from those of TokenScript. While TokenScript outlines how smart tokens should be packaged, distributed, and executed, Smart Layer emphasises the real-time execution of the server-side components of these scripts. This ensures they function optimally within the defined parameters of privacy, secure storage, and cost accounting. The features specific to TokenScript, currently under standardisation in collaboration with OASIS, are not covered in this document.

Smart Layer Architecture

Smart Layer's architecture is rooted in mature protocols and algorithms that have been proven effective in distributed systems, including blockchain itself, distributed hash table, load balancing and service level objective monitoring, and the use of Merkle tree in data integrity verification. These foundational technologies provide the basis for building Smart Layer as a robust, decentralised network tailored for token operations. The innovation primarily stems from the creation of an integration service platform and a conducive environment for smart tokens, encouraging existing web infrastructure to transition towards a token-centric architecture.

The primary serviceability requirement determined that the network cannot be built like a blockchain, where consensus serves to determine truth; instead, services must be monitored and load-balanced in real time. This leads to the need for anchoring nodes.

Anchoring Nodes and Distributed Smart Token Instances

Smart Layer's emphasis on serviceability sets it apart from traditional blockchains that lean heavily on consensus mechanisms. This focus demands real-time monitoring and load balancing, which is where anchoring nodes come into play. These nodes serve as the network's guardians, ensuring consistent service availability and stepping in for pivotal operations. The Distributed Hash Table (DHT), shared among these anchoring nodes, is instrumental in determining which node is responsible for a specific smart token instance. This decentralised approach not only mitigates potential attacks that might arise from matching node IDs with token IDs but also guarantees prompt responses to integration queries.

An integration, which is to say, a website and its browser session combined, access anchoring nodes to get access to the service nodes of the token the user decided to use in that integration.

The anchoring nodes, working with their peers, maintain a distributed hash table to find service nodes for any given smart token.

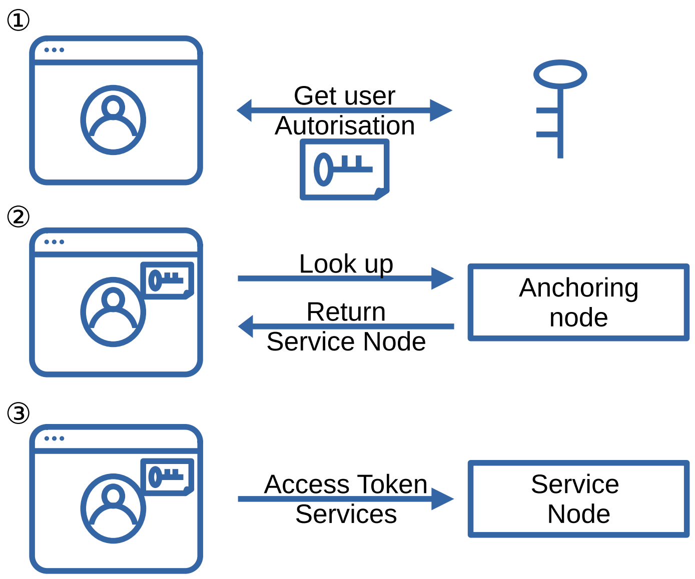

Figure 3: Process of getting to the service node

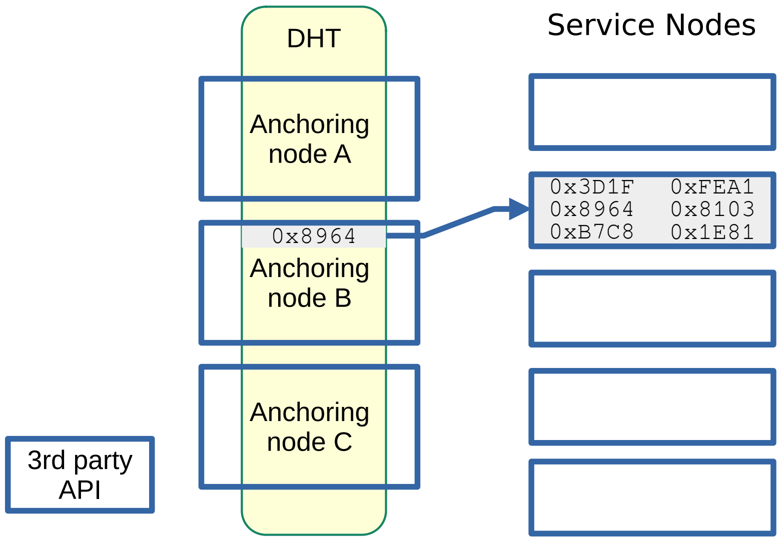

Figure 4: Mapping Token ID to its service node

Token Status Propagation and Execution

Smart tokens, as envisioned in the Smart Layer network, have a dynamic status that can be influenced by various factors. These factors can range from attestations to smart contract updates and node messages. While some of these updates are deterministic, others can be non-deterministic, leading to potential complexities in the network's operation.

Deterministic vs. Non-Deterministic Status

A deterministic status update is one that, given the same input, will always produce the same outcome. For instance, with a flight ticket as a smart token, a flight delay leading to an automatic lounge access reward for a passenger is deterministic. However, not all updates are so straightforward. Consider the scenario of the same smart token rebooking a hotel through a web API. The outcome might be a successful booking attestation, a timeout, or even a server-side error. Such non-deterministic outcomes present challenges, especially when integrating with existing Web2 systems. While there are pure blockchain-based solutions that completely do away with the status branching, such as rebooking through hotel smart contracts backed by a hotel's precommitment, the integration of web2 systems with smart contract-enabled platforms will remain a challenge for the foreseeable decade.

Execution Verification

In the Smart Layer network, the primary objective of execution verification is to ensure the integrity of the services provided to integrations. It's crucial to understand that this verification process is not a duplicate of validating the execution of the core logic of the smart token contracts. Instead, it focuses on the accuracy and trustworthiness of providing integration services to integrated systems.

To illustrate with a real-world smart token use-case, imagine a smart car that breaks down on the road, and a smart insurance token is at work. Execution verification ensures that the driving data is accurately passed to the roadside assistance company. It doesn't concern itself with whether the insurance payout due to the breakdown is calculated correctly - such core token logic can be part of the insurance Smart Contract, and its trust is derived from the underlying blockchain. Consequently, a potential exploit in the execution verification of any Service TokenScript would more likely target integrated systems rather than attempt to manipulate smart contract payouts.

With this context in mind, anchoring nodes in the Smart Layer network are tasked with verifying the execution of Service SmartTokens. This introduces several protocol requirements:

1. Inputs to the Service TokenScripts should be structured as valid attestations and safeguarded against replay attacks and misuse in inappropriate

contexts.

2. Failures of service nodes in obtaining such attestations must be verified by anchoring nodes to prevent false claims of failures of the integrated systems.
3. The output resulting from token execution should be attested, offering proof of the operation's authenticity.

Let's delve into these requirements:

Inputs must be attestations

To bolster the integrity and authenticity of data inputs, they should be framed as attestations. These cryptographic proofs validate the legitimacy of specific data or actions. Mandating inputs as attestations ensures that only authenticated and verified data drives token operations, enhancing the security and reliability of the Smart Layer network.

Handling Failures

When a node falters in its execution responsibilities, anchoring nodes intervene. They can either mediate disputes or reassign the smart token to a more dependable node. Only anchoring nodes can issue attested failures. Their primary role is to provide attestation for the failure to secure necessary attestations, propelling the token to its subsequent state.

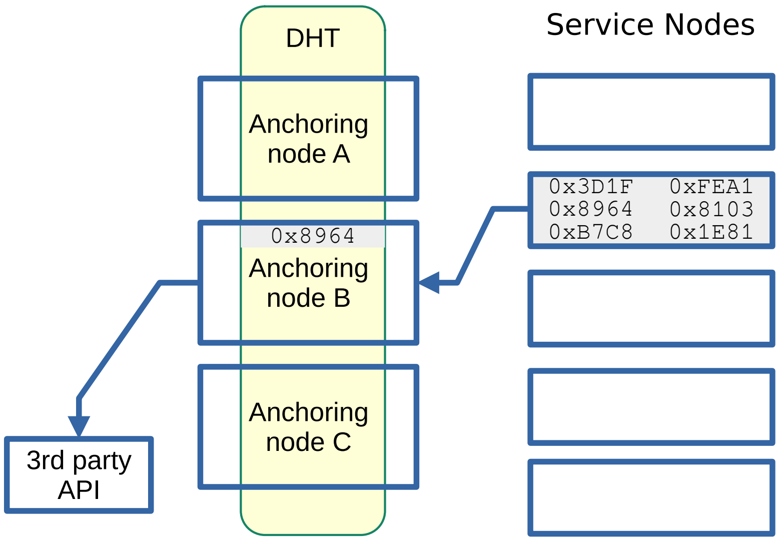

Figure 5: After a failure, a service node requests an anchoring node to route its traffic in order to get a failure attestation

Attested Execution

Trusted Execution Environments (TEEs), such as Intel's SGX, offer a secure milieu for code execution, safeguarding data confidentiality and integrity. Leveraging TEEs facilitates attested executions, where computation results are paired with proof of correct execution.

Service Nodes must execute all Service TokenScript within TEEs verifiable by anchoring nodes and integrations. This stands as our primary mode of execution verification.

Security While TEEs, including SGX, have encountered vulnerabilities, they remain unparalleled in ensuring trusted execution without overburdening the design with consensus protocols. Large commercial entities, like Microsoft's Azure, have embraced TEEs, reinforcing confidence in the technology. However, the system should be equipped with alternative execution verification fallbacks in case of vulnerabilities. These fallbacks, activated by a DAO emergency vote, range from partial (load-balancing nodes across different platforms) to full (routing all service node traffic to anchoring nodes for selective computation verification).

Performance TEEs, such as SGX, have limitations on computational power utilisation. In practice, node operators might offset this by running parallel tasks like mining. Future TEE iterations aim to optimise resource allocation and advancements like parallel execution, and Enhanced Memory Management promises near-full system resource utilisation for Service TokenScript execution.

Periodic Execution Monitoring

Anchoring nodes are mandated to periodically oversee the execution of Service SmartTokens. This continuous monitoring regulates the staking mechanism, allowing for stake slashing upon detecting execution discrepancies. Service nodes are periodically prompted to provide execution samples for validation. This mechanism is akin to immune cells inspecting protein synthesis within biological cells, ensuring operational integrity.

In fallback mode, execution monitoring is performed by selectively redoing the execution, relying on all service nodes to route traffic to anchoring nodes, effectively transforming them into gateways.

Real-World Application: The Flight Ticket Smart Token

To better understand the intricacies of the Smart Layer network, let's delve into a real-world example: the flight ticket smart token.

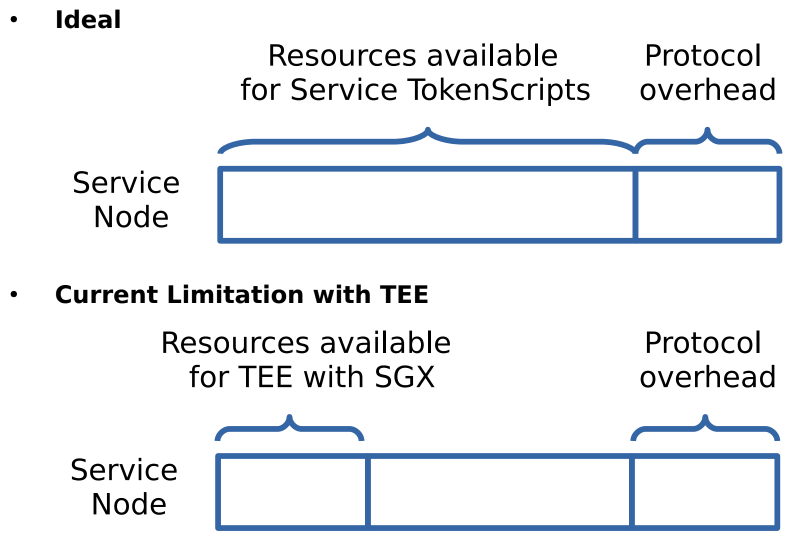
Ideal
Resources available for Service TokenScripts
Protocol overhead
Service Node
Current Limitation with TEE
Resources available for TEE with SGX
Protocol overhead
Service Node

Figure 6: TEE negatively affect the performance of nodes, however, the force of economy may lead nodes to allocate resources to other services

Read-Only Access

Imagine the user booked a car rental at the rental website, using the smart token on this website. This creates an authorisation for the car rental to access the flight ticket smart token interfaces. When the car rental service wants to verify if your flight is on time, any node that has this smart token instance can provide a read-only API to share the flight's current status, and the selection of the node is largely a matter of load balancing. This process is facilitated by any of the nodes where the smart token is instantiated and selected at random.

Single Execution by Elected Node

Now, consider a significant flight delay. This delay might trigger the smart token to rebook your hotel reservation. Since this action interacts with the external world and can have financial implications, it is crucial that only one node executes it. Once the rebooking is successful, the hotel system might generate a new booking attestation, confirming the change.

However, what if there's a failure? What if the node trying to rebook the hotel faces multiple timeouts when accessing the hotel's API? After a set number of failures, from the network point of view, an event occurred that the token failed to move to the next state. The traffic to the hotel system API would be rerouted through the anchoring node that elected the original executor. This anchoring node can then verify server timeouts or other errors and provide a failure attestation to move the smart token to the next state. Frequent routing of this nature triggers service level agreement to scrutinise the node that was responsible for execution.

While the flight ticket smart token serves as a tangible example, it is essential to understand that the Smart Layer network is not limited to this application. The principles discussed here apply to a myriad of potential smart tokens, each with its unique challenges and solutions.

Attestation Gossiping and Queuing in Smart Layer

In the Smart Layer architecture, the dynamic status of smart tokens is predominantly updated through attestations. These attestations, essentially cryptographic proofs, vouch for the validity of a particular state or action. Given the decentralised nature of the system and the real-time requirements for token status updates, there arises a need for an efficient mechanism to disseminate these attestations across the network.

The Need for a Hybrid Mechanism

Traditional gossip protocols, inspired by the way information (or gossip) spreads in social networks, have been a staple in distributed systems. They ensure that data is disseminated quickly and efficiently across a network. However, the unique requirements of Smart Layer, especially the need for ordered delivery of attestations and the ability to request missing attestations, demand a more sophisticated approach.

The unique requirements include selective gossiping where an attestation covers a type of smart tokens or a subset of smart tokens in that type, the timely update needed for integration to function, order and integrity, plus compatibility with a subscription model. This calls for a hybrid mechanism, which marries the strengths of gossip protocols with the features of systems like Apache Kafka. While gossip protocols ensure rapid dissemination, systems akin to Apache Kafka ensure that these attestations are delivered in order and allow nodes to request specific attestations they might have missed.

How It Might Work

Imagine a scenario where an airline releases a series of attestations updating flight arrival times. Nodes in the Smart Layer network that have smart token instances depending on that flight would subscribe to these attestations. As these attestations are gossiped through the network, the Kafka-like system ensures they are received in the correct order. If a node misses an attestation, it can request that specific piece of information, ensuring data integrity and consistency across the network.

This hybrid approach addresses the challenges of both rapid dissemination and data consistency. The decentralised nature of Smart Layer presents unique challenges, and we are committed to innovating and iterating on these solutions to ensure a robust and efficient system. As we progress, we will continue to refine and adapt our approach to best serve the needs of the Smart Layer ecosystem.

Secure Execution Environment

Given the dynamic nature of smart tokens, it is imperative to have a secure environment for executing token code. Sandboxing techniques are employed within Smart Layer nodes, allowing token scripts to run in isolated environments. This ensures that the broader network remains unaffected by potentially malicious or faulty token scripts.

Smart Layer Token Types

The tokenomics of Smart Layer is intricately designed to ensure the sustainability, efficiency, and robustness of the network. Central to this design is the dual-token system, which serves distinct yet interconnected purposes:

Service Token

The Service Token is the primary medium of exchange within the Smart Layer network. It facilitates all micro-transactions associated with token operations, from querying token data to more resource-intensive modifications. Integrations typically bear the costs associated with token activation, which are transacted using the Service Token. This token ensures that the network remains agile, with real-time settlements and efficient resource allocation.

Stake Token

The Stake Token represents a stake in the Smart Layer network. Holders of this token have a vested interest in the network's growth and governance. Depending on the network's design, Stake Token holders might influence governance decisions, propose changes, or even earn rewards based on network activity. This token ensures that the network remains decentralised, with stakeholders actively participating in its evolution.

The tokenomics of Smart Layer is designed to ensure the sustainability, efficiency, and robustness of the network. It strikes a balance between incentivising token issuers, integrations, and the nodes that power the network. Here's a deep dive into the tokenomics structure:

Service Token’s Tokenomics

Token Issuer Rent

Every token issuer pays a nominal fee termed as “rent.” This rent ensures that the TokenScript associated with the token remains available on the network. Additionally, it guarantees that any smart contracts linked to the token are actively monitored. While the rent is minimal, it serves a crucial purpose: it ensures that the network remains primed for any token activations, ensuring the readiness and responsiveness of the Smart Layer.

However, this rent does not necessarily translate to a financial burden for token issuers. On the contrary, they can potentially profit from the broader ecosystem, particularly from the “business” operations, which will be elaborated upon subsequently.

Token Activation and Operation Costs

While the token issuer pays the rent, the costs associated with token activation are typically borne by the integrations. An integration, in this context, refers to any stakeholder or entity that leverages the smart token for a specific use-case.

For instance, consider a health token. An e-commerce platform might utilise this token to optimise a user’s shopping cart. In such a scenario, the e-commerce platform would cover the costs associated with accessing the token’s interface. Conversely, a visa office might adopt a different approach. If an applicant provides data, such as a flight ticket in smart token format, the visa office might require the applicant to attach a small payment to cover the token operation costs.

The cost of querying token data is relatively low, facilitated through a state channel. However, modifying token data, given its reliance on gossiping and node synchronisation, is more resource-intensive and costlier.

State Channels and Settlement

State channels play a pivotal role in the Smart Layer tokenomics. Whenever an integration interacts with a smart token, a state-channel is opened. This channel keeps track of all the micro-transactions associated with the token operations. Integrations have the flexibility to close these channels and settle the accumulated costs at regular intervals, such as monthly.

Anchoring nodes, which play a central role in the network’s operation, maintain these state channels. They ensure that payments are routed correctly to the nodes that provide the services. At the end of a settlement period, the anchoring node consolidates all the transactions within the state channel and processes the payment to the respective service-providing nodes.

Dynamic Tokenomics: Tailoring Incentives in a Decentralised Ecosystem

The tokenomics is structured in a way that allows token issuers, while paying a nominal rent, to write their token contract in a way that profits from the broader ecosystem.

Token contracts can dictate revenue derived from various "business" operations facilitated by the smart tokens. For instance, a token issuer might collaborate with multiple integrations, each offering a unique service or benefit associated with the token. Every interaction with the token, whether it is a query or an update, translates to a micro-transaction. This is because unlike industrial smart tokens, some community smart tokens may need the cumulative value of these micro-transactions.

Furthermore, integrations, by leveraging smart tokens, can offer enhanced services to their users. This improves user experience and may open up new revenue streams for the integrations. For instance, an e-commerce platform can offer personalised shopping recommendations based on a user's health token, leading to increased sales and customer satisfaction.

Relationship with Other Projects

IPFS

Smart Layer's architecture is inherently designed to be modular and interoperable, a philosophy that aligns with the InterPlanetary File System (IPFS). While IPFS serves as a decentralized storage layer, it focuses primarily on content availability without service-level guarantees such as I/O and response time.

This positions IPFS more as a retrieval service than a web service, lacking a Virtual Machine (VM) for code execution.

Smart Layer offers the option to use IPFS for storage, particularly when the higher costs associated with Smart Layer's features like load balancing and Service Level Agreements (SLAs) become a concern. This strategic alignment allows Smart Layer to maintain its lightweight nature while ensuring data integrity and availability, key attributes that are indispensable for any decentralized application (dApp).

Chainlink

Smart Layer's collaboration with Chainlink significantly enhances its capabilities, particularly in the area of decentralized oracles. Chainlink is renowned for its decentralized oracle services, which provide secure and reliable data feeds to smart contracts. However, Chainlink's primary focus is on linking external data to blockchain environments, rather than enabling smart tokens as integration points. It does not concern itself with providing token interfaces for specific use-cases like smart flight tickets.

On the other hand, Smart Layer adopts a token-oriented approach, where smart contracts act as trust anchors. This architecture allows for a separation between rule enforcement and execution logic, offering greater flexibility in application development. By integrating Chainlink's robust oracle services, Smart Layer can access real-world data, thereby enabling more complex smart contracts that can interact with external APIs, IoT devices, and other data sources. This synergistic relationship broadens the scope of applications that can be built on Smart Layer, ranging from decentralized finance (DeFi) to supply chain management and beyond, while also providing the necessary infrastructure for specialized token interfaces.

This distinction in scope allows each platform to excel in its area of expertise, while their integration offers a more comprehensive and versatile solution for decentralized applications.

TokenScript

TokenScript is a direct dependency of the Smart Layer technology stack. The same team that has been instrumental in the development of TokenScript has also been responsible for Smart Layer. This team's work on TokenScript has been recognized by the OASIS Standardization body as part of its collaboration with the Ethereum Foundation. While Smart Layer aims to provide a robust integration infrastructure for the next generation web, TokenScript focuses on standardizing token interfaces, behavior code, and attestation mechanisms. These elements are essential dependencies for the Smart Layer network.

The integration of TokenScript into Smart Layer is a fundamental requirement for the latter's operation. TokenScript's XML-based token markup language enables a modular approach to dependency-based token interoperabilities. For example, it allows the insurance industry to establish standardized interfaces for smart insurance tokens. Utilizing TokenScript's editors and deployment tools, developers can define token behavior without writing XML directly, using familiar languages like JavaScript to control a token's runtime, whether in a wallet or within the Smart Layer network.

Conclusion

In conclusion, Smart Layer integrates IPFS as an optional storage solution, benefiting from its mature implementations of distributed hash tables, but does not rely on it for runtime operations. Chainlink serves as a source of attestations and is an optional component, the utilization of which is contingent upon specific smart token authorizations. Unlike IPFS and Chainlink, TokenScript is not an infrastructure but a standard for defining smart tokens. It serves as a direct dependency, essential for the functionality and interoperability within the Smart

Layer ecosystem.

Design Considerations and Summary

Designing for today's unthinkable, tomorrow's norm

As we delve into the intricacies of protocol design, it's crucial to remember that today's innovations often become tomorrow's standards. The Smart Token and its supporting Smart Layer are designed with this forward-thinking approach. For example, consider the potential for future retailers to interface with vehicle smart tokens for predictive maintenance or flight smart tokens for timely deliveries.

The protocol is also designed to accommodate emerging scenarios, such as smart locks granting access based on tokenized rights or smart cars initiating autonomous roadside assistance. Furthermore, as AI becomes increasingly integrated into decision-making, the protocol is engineered to support AI-driven decisions through token interfaces. This design choice enhances security and composability, essential attributes for future web integrations.

The Decentralised Nature of Future Integrations

Contrary to the notion that the protocol's success depends solely on adoption by Internet giants, the true power lies in its ability to connect various stakeholders. The next-generation Web will likely be a mosaic of localized innovations tailored to specific industries and users. In this context, Smart Layer aims to serve as a flexible and adaptable foundation.

The protocol emphasizes a layered design approach, focusing on provisioning smart tokens as a robust mechanism. This allows for the development of more complex features and applications atop this foundational layer, without getting entangled in the specifics of individual tokens.

Summary and Implications

In this paper, we have presented Smart Layer as a protocol designed for decentralized integration in the next-generation Web. The protocol leverages smart tokens to facilitate interactions between various web services, sidestepping the need for centralized entities. Unlike tokenized assets, which are primarily designed for trading, smart tokens in this protocol are engineered for specific applications. This focus aligns with the evolving technological landscape and its emerging use-cases.

As the paper concludes, it's worth noting that the protocol is not a static entity but a continually evolving framework. It aims to contribute to a more interconnected web ecosystem, and as such, invites ongoing engagement from developers and stakeholders for its further refinement and expansion.