Smart Token: The Building Block for the Next-Generation Token-Centric Web

1st October, 2023

Abstract

The evolution of the Web has been marked by significant shifts, from Web 1.0's flat architecture, where the Web was primarily seen as an information repository akin to books, to Web 2.0, where the Web transformed into an application platform. This transformation led to a "reverse pyramid" structure, with contemporary internet behemoths forming the narrow, foundational base. Such centralisation has stifled the Web's innovative potential. As the number of users and websites has surged, the past decade has witnessed a plateau in transformative platforms or groundbreaking innovations, with the digital terrain largely commandeered by a few familiar giants. This paper delves into the root causes of this innovation drought, emphasising the indispensable role of trust anchors in nurturing a vibrant web ecosystem. We introduce the concept of a Token-Centric Web, a vision for the next-generation Internet that decentralises trust and enables an ecosystem of integrations. In this context, we propose "Smart Tokens," leveraging smart contracts, as an architectural choice to instantiate trust anchors in this Token-Centric Web to amplify user experience, bolster privacy, reduce dependence on monolithic Internet titans, and foster a new wave of web innovation. The paper further probes the potential for transformative shifts across various web dimensions and delineates the technical challenges, potential pitfalls, and adoption hurdles.

Contents

Web Foundations and Trust Anchors **2**
- The Web's Foundational Model 2
- The Shift from Information to Applications 3
- Trust Anchors and Web Centralisation: An Unintended Byproduct of Application Shift 4
- Case Study: OpenBanking 4
- Case Study: Google Login 4
- The Limit in the provision of trust Anchor leads to centralisation and innovation barrier 5
- Case Study: Google Pay and Google Wallet 6

Reimagining Web 3.0: Beyond the Internet of Value 7

The Token-Centric Web: Decentralised Trust and Integration 8
Missed Opportunities of Decentralised Trust Anchors 8
Overcoming Barriers to Competition 8
Facilitating Previously Impossible Innovations 8
Enabling Long-tail Trust Anchors 9
Envisioning the next-generation Web 9

Smart Tokens: The New Trust Anchors 9
Token as the Trust Anchor 9
Premise 1: Trust is on the Tokens, not on the Platforms 10
Premise 2: Using Token for trust anchor Allows for Layered Design 10
Transitioning from Smart Contract Tokens to Smart Tokens 11
Smart Token Architecture 13
Requirements of Token Runtime Environment 13
A Tripartite Approach 16

Web Foundations and Trust Anchors

The Web's Foundational Model

When Tim Berners-Lee and his team developed the foundational concepts of the Web, they selected "sites" as its primary building blocks. This seemingly intuitive approach, however, was not a given, especially when other Internet protocols, like emails and USENET, did not revolve around the concept of sites. Consider USENET: it organises and manages information by topic, making it irrelevant which site or even which planet the information originates from under that topic.

The Web embraced a site-centric model: a site has a single origin, is inherently competitive, forms an ecosystem through hyperlinks, and evolves as a platform rather than a static product. This model complemented the revolutionary capabilities introduced by HTML, contributing to the Web's rapid adoption and eventual dominance as an Internet application. Even in the mobile Internet era, the foundational model of 'sites' persisted[1].

Berners-Lee and other early web pioneers didn't adopt the "site" concept merely for its potential evolutionary power. Instead, the Web's design was heavily influenced by a prevailing metaphor of that era — the library model, which likened the Internet to a vast library. This metaphor transposed a library's

[1]Although early mobile system designers envisioned apps to be function-centric, akin to desktop word processors and movie players, the reality differed. Instead, users embraced mainstream apps like Google Docs and Netflix. Like a site, a mainstream mobile app possesses a single origin, thrives in competition, links to other apps, and remains open to ongoing development. This evolution is a testament to the enduring influence of the site-based model, even in a landscape that has shifted significantly from Berners-Lee's original vision of the Web.

concept—a collection of books—to the digital realm, turning the Internet into a collection of sites. Just as a book references pages, the Web adopted "web pages." This framework led to structuring the Internet around origins (sites) instead of topics (as in USENET) or functionality (as in FTP). Hyperlinks became akin to library indexes, but site owners controlled these links, creating a self-referential mega-book that spanned the entire library.

Structuring the Web around origins rather than topics (like USENET) or functionality (like BitTorrent) had vast implications. It led to websites' single-origin design, reminiscent of how books have specific authors. Today, multi-domain sites are rare. This design choice profoundly shaped our trust paradigm: we often interact with a site based on the trust we place in its origin. A site isn't just an information repository; it represents its origin's credibility.

This decision, to be explored further in subsequent sections, inadvertently paved the way for the Web's centralisation.

The Shift from Information to Applications

Originally influenced by the metaphor of a universal library, the Web was conceived as an information system[2]. Today, such a description feels outdated. A more fitting depiction of the modern Internet is a sprawling network of web applications. Rarely do individuals now describe their online activity as simply "browsing" for information. Instead, they're interacting with dynamic web apps to chat, shop, book hotels, work remotely, network, or for leisure activities. Few draw parallels between the Internet and a Universal Library these days.

The transition from a web of information to a web of applications marked the significant transformation of Web 2.0. This was achieved by expanding the site model into an application model through web services. Key technologies of Web 2.0 include AJAX, RESTful API, and SaaS. Notably, these are application-oriented technologies built atop the traditional site-based information model. Concurrently, HTML evolved from a document format to an application development User Interface description language.

As the Web transitioned from an information repository to an application platform, the concept of trust anchors emerged. These trust anchors - essential services the web ecosystem depends on, are usually provided by a few dominant entities and become the focal points of trust for web applications. This dynamic has inadvertently led to the centralisation of the Web, stifling innovation and competition.

[2]Its USENET topic name comp.infosystems.www accurately captured it.

Trust Anchors and Web Centralisation: An Unintended Byproduct of Application Shift

The advent of Web 2.0 has given birth to an intricate network of applications, each interconnected and reliant on a multitude of web services. A typical mainstream website might incorporate as many as 10 to 15 mainstream web services, such as Google Login and Google Pay. Unlike traditional applications that depend on system components, these web services have evolved beyond their functional roles, becoming custodians of trust that ensure secure and reliable operation of web applications. In this paper, we refer to these pivotal web services as Trust Anchors.

In this section, we will dissect the concept of trust anchors, explore their role in the centralisation of the Web, and propose a decentralised alternative through the implementation of Smart Tokens.

Defining Trust Anchors Trust Anchors are external web services that web applications depend on for their core functions, even when they possess the code to provide similar functionalities themselves. This dependency arises from the inherent trust that the business logic of these applications places on these services. In essence, Trust Anchors are not just functional, they are trusted.

The crucial distinction between a website's functionality and the trust it depends upon is demonstrated by the following Case Studies:

Case Study: OpenBanking

To illustrate this point, let's reflect on a case from 2008. An author of this paper, while working with the German Chamber of Commerce in China, conceptualised a website named 'OpenBanking'. OpenBanking aimed to offer superior functionality to existing banking platforms by integrating with the popular Home Banking Computer Interface, providing users with features not available on their banking platforms, such as expenditure pie charts.

Despite its open-source nature and advanced features, users were hesitant to input their credentials into OpenBanking. Interestingly, these same users had no issues using similar functionalities on open-source desktop software. This paradox underscores the importance of trust anchors in user adoption. In this case, the recognised trust anchor was the German bank, and without its explicit endorsement, users were reluctant to trust OpenBanking.

This case study highlights the challenges in establishing new trust anchors and strengthens our argument for the importance of decentralised trust anchors.

Case Study: Google Login

When a user attempts to authenticate with a website, two challenges arise. Firstly, users often doubt the website's ability to safeguard their data. Secondly,

the website may struggle to verify the authenticity of the user. This predicament often compels users to prefer logging in with Google, a trusted entity, and websites to adopt Google's authentication service. Currently, the reliance on social logins has grown to such an extent that some websites have entirely eliminated traditional password databases.

In this scenario, Google Login serves as a trust anchor, providing functionality and acting as a beacon of trust. Users place their faith in Google to authenticate genuine users without compromising or inadvertently leaking login credentials. This creates a dependency on a central point.

While open authentication protocols like OAuth and OpenID allow any website to implement a secure and reliable authentication system similar to Google Login, they typically lack the trust factor necessary to become a trust anchor. A new entity, unlike Google, may not be trusted to maintain consistent behaviour, as it could potentially be compromised, deviate from protocol, or cease operations.

The case studies above help us to better understand the characteristics of Trust Anchors.

Characteristics of Trust Anchors

1. A trust anchor is an essential dependency, not just a popular service. For example, Google Analytics is not a trust anchor, as a website can carry out its own analysis. Its popularity is based on traditional factors of centralisation like big data, network effect, and IP laws, not on its role as a trust anchor.
2. A trust anchor is trusted to follow a protocol and is expected to not betray the trust.
3. A trust anchor elevate user trust to meet the demand of the application. For instance, users willingly provide their credit card details to Google Pay for transactions on merchant websites. However, not all service providers act as trust anchors. For example, Amadeus, which provides flight data and reservation services for airlines, operates in the airline website's back office without needing to elevate user trust.

The Limit in the provision of trust Anchor leads to centralisation and innovation barrier

Dr. Gavin Wood has attributed the Web's centralisation to a combination of factors. These factors include network effects, economies of scale, big data ownership, and intellectual property laws[3].

This paper posits that the role of trust anchors significantly amplifies these factors, coalescing them into a formidable force that reinforces the centralised dominance of today's tech giants. These entities, including Facebook, Google, and Apple, derive their trustworthiness and operational integrity largely from

[3]Wood, "The Future of the Decentralized Web."

their scale and profitability. The rationale is that such entities have amassed substantial profits by being reliable providers of trust anchors. As a result, any deviation from their established behaviour for short-term gain is considered economically irrational.

Such dynamics have exacerbated the centralisation within the Web 2.0 ecosystem, culminating in an oligopolistic Web 2.0 space.

The trust anchors by the Internet centres, once formed, creates an innovation headwind.

The Trust Anchor Effect: Innovation Stifled by Centralisation We define the "Trust Anchor Effect" as the phenomenon where the centralisation of trust within a few dominant entities creates a significant barrier to innovation. This effect describes a web ecosystem where new products and services, despite being technically feasible, remain unrealised due to the absence of trust in entities other than the established trust anchors. It encapsulates the dependency on these central points for the provision of trust, without which innovation cannot gain traction or user acceptance.

The trust anchor effect is evident in scenarios where a web service's ability to innovate is contingent upon the trust anchors' willingness or readiness to support new functionalities.

Case Study: Google Pay and Google Wallet

Google Pay, when integrated into web platforms, enables users to complete transactions without directly exposing their credit card details to the merchant's website. Serving as a Trust Anchor, Google Pay extends beyond mere transactional functionality; it is entrusted with ensuring reliability and operational integrity. Even if an open-source developer were to create a feature-wise superior payment system, it lack the level of trust that to function as a Trust Anchor.

With the evolving demands of e-commerce, Google rebranded Google Pay to Google Wallet, expanding its capabilities to store not only credit cards but also items like shopping vouchers and digital car keys. However, these are not made into Trust Anchors.

For example, a website that accepts the shopping vouchers during the checkout process can't use the voucher stored in Google Wallet directly, despite it could with credit cards in Google Wallet. User is required to copy and paste the voucher code, as Google Wallet has yet to develop the voucher as a Trust Anchor service. Similarly, although a user can store a digital car key in Google Wallet, this does not extend to allowing a car cleaning service's website to access the car for service purposes. The user still need to carry a physical car key at the cleaning appointment.

This means any web innovation built on top of the recognition of the shopping voucher and use of digital car key cannot proceed unless Google developed them

into Trust Anchor services, creating an innovation dependency.

In essence, the trajectory of Web 2.0 innovation is not solely constrained by the technical ingenuity of developers but is significantly influenced by the strategic priorities of the incumbent Internet powerhouses. The current ecosystem operates under a paradigm where new entrants are beholden to the established trust anchors, which act as gatekeepers of progress. This dynamic has led to a web landscape that, while ostensibly advancing under the leadership of tech giants, is in fact characterised by a latent inertia. Innovators find themselves in a position analogous to infantry in an army, where their advance is not limited by their own capabilities but by the strategic decisions of the commanding officers. The result is a web environment that is less a meritocracy of ideas and more a hierarchy of trust, with innovation potential tethered to the discretion of a few dominant entities.

Reimagining Web 3.0: Beyond the Internet of Value

The prevailing vision for Web 3.0 is characterised as an 'Internet of Value' - a platform where value, in its various forms, is distributed more equitably among users and creators, breaking away from the monopolistic tendencies of the Web 2.0 era. The centralisation of value, in this view, is seen as the root cause of many of the issues plaguing the current web ecosystem, from stifled innovation to privacy concerns.

However, this paper proposes a different interpretation of the evolution from Web 2.0 to the next generation web after it. Rather than viewing the centralisation of value as the *cause* of the Web's current evolutionary obstacles, we argue that it is, in fact, a *consequence* of the Web's evolution. The transition from Web 1.0 to Web 2.0 was not primarily driven by a pursuit of centralised value, but by the demand for more dynamic, application-oriented experiences. Therefore, it is not logical to assume that the transition to Web 3.0 should be defined by a reversal of this trend.

Furthermore, the characterization of Web 3.0 as an 'Internet of Value' may be more reflective of a countercultural movement against the early elitism Web 2.0, spurred by social and economic disparities, rather than an evolutionary process of the Web. Not everyone who transitioned from prior-generation Web to Web 2.0 was pursuing value, and not everyone who migrates from Web 2.0 to the next-generation web will necessarily be pursuing value either. While the value investing community is growing, it still represents a small section of web users and is likely to remain so in the future.

Instead of defining Web 3.0 as an 'Internet of Value,' this paper proposes an alternative vision: the 'Next-Generation Web.' This Next-Generation Web shifts focus from the distribution of value to the decentralisation of Trust Anchors, attacking the root cause of centralisation while continuing to progress in the demand-driven direction that has been driving the Web 1.0 to 2.0 upgrade. This perspective views the Next-Generation Web as a platform that fosters widespread

innovation and enables more profound integrations by allowing anyone to develop and maintain Trust Anchors.

The Trust Anchors of the future Web will not be platforms or services, but tokens - specifically, Smart Tokens. The ensuing sections will delve into this concept in more detail.

The Token-Centric Web: Decentralised Trust and Integration

The next-generation Web, as envisioned in this paper, represents a paradigm shift from the centralised trust anchors of today to a decentralised and integrated ecosystem. This transformation is predicated on the ability to establish trust anchors independently of internet giants, thereby democratising the Web's trust infrastructure. In this chapter, we will explore the implications of this shift, the areas it would revolutionise, and how it culminates in a web defined by limitless integration.

In the current web ecosystem, trust anchors are the domain of a few centralised entities, which has led to a web that is both siloed and constrained by the strategic priorities of these entities. By allowing anyone to develop and maintain trust anchors, the gatekeepers could be removed to enable a web that is more resilient, diverse, and conducive to innovation.

With the removal of centralised control over trust anchors, web services would no longer be limited to integrating a narrow set of core functionalities. Instead, they could leverage a wide array of trust anchors tailored to their specific needs. This would lead to a seamless and cohesive user experience, as services could integrate more deeply with one another.

Missed Opportunities of Decentralised Trust Anchors

Overcoming Barriers to Competition

Missed Opportunity: By removing the monopoly over trust anchors, entities that previously hoarded valuable reputation data would no longer serve as gatekeepers. This would enable a more fluid market where reputational capital can be a portable asset, fostering a more dynamic and competitive landscape.

Example: E-commerce platforms could benefit from a decentralised system where a seller's reputation and customer reviews are not confined to a single platform. This would allow sellers to utilise their established reputation to gain financing or expand their business across various marketplaces.

Facilitating Previously Impossible Innovations

Missed Opportunity: With a decentralised trust anchor system, new services that rely on the integration of multiple trust anchors could emerge. These

services would be able to offer highly personalised and flexible experiences that adapt to changing user needs and contexts.

Example: A personalised travel guide service could leverage trust anchors to seamlessly manage and adjust travel plans, including bookings, accommodations, and activities, based on the user's real-time preferences and circumstances. This level of integration and flexibility is unattainable in the current centralised trust anchor environment.

Enabling Long-tail Trust Anchors

Missed Opportunity: Decentralised trust anchors would allow smaller entities to establish their own credibility mechanisms. This could lead to a proliferation of niche platforms that can cater to specific community needs or specialised markets without the need for endorsement from large internet giants.

Example: In the car insurance industry, small insurers could use decentralised trust anchors to validate car ownership, driver identity, and maintenance records without relying on cumbersome paper processes. This could streamline operations and allow them to offer competitive rates and services.

Envisioning the next-generation Web

In consideration of the implications previously discussed, this paper proposes a conceptual framework for the next-generation Web. This envisioned Web is distinguished by the proliferation of ubiquitous trust anchors, which enable limitless and profound integrations across services and platforms. Such a web facilitates a user experience that surpasses the capabilities of the Web 2.0 era and drives innovation forward to enable types of sites and services that couldn't exist prior.

Notably, this vision diverges from the popular concept of Web 3.0, which is characterised as an 'Internet of Value.' Instead, our focus is on the transformative potential of integrations made possible by accessible and universal trust anchors.

Smart Tokens: The New Trust Anchors

The preceding chapter outlined a vision for a web populated by ubiquitous trust anchors. This chapter posits that these trust anchors should assume the form of tokens. We will explore why tokens are suitable as trust anchors, discuss the technical and layered design implications, and introduce a new design requirement for the type of tokens suitable for trust anchors: smart tokens.

Token as the Trust Anchor

Reflecting on the case of Google Pay/Google Wallet as a trust anchor, one might envision a decentralised trust anchor as a similar entity, such as a hypothetical

"DecentWallet." However, this paper argues that the trust anchors should be tokens.

This argument rests on two main premises: one concerning trust and the other concerning layered design.

Premise 1: Trust is on the Tokens, not on the Platforms

Firstly, regarding trust, we argue that tokens, not software or platforms, are the actual focal points of trust dependency.

Consider previous examples, such as a car key token. If implemented as a trust anchor, it could enable many innovative use-cases. Naturally, questions arise: Should a car wash website accept any car key token authorisation and allow the car owner to park the car and walk away? Since many Google APIs are open, one could sign up, create a token called "MyCar," generate a key, and use that on the reservation webpage. The car wash website would accept a non-existent car. Some form of validation must take place.

Two potential solutions arise: the website could maintain a trusted car key token list, or Google could become the gatekeeper, not allowing production car key token releases unless the developer can prove that they are coding the car key token for a genuine car manufacturer. The latter is more practical, so Google becomes not only the car key web service provider but also a curator of valid car key token lists.

However, Google cannot ensure that the car key token functions as the web applications depending on them expect. Each car key token is programmed by their respective car manufacturer, and Google is not in a position to audit their code. Ultimately, the trust lies with the token itself, and Google acts as a transferer of trust rather than the origin of trust. The car wash website essentially trusts that if Google recognises the car token as being genuinely programmed by a car vendor, such as Tesla, then Tesla would not program their car key token to create invalid authorisations just to spoof the website.

With public key cryptography, it is not a problem to attest that a car key token is programmed by Tesla. Therefore, Google's role is reduced to a curator, and trust remains with the issuer of the car key token, such as Tesla. What makes Google a better curator of valid car keys? They don't produce any cars, nor do they own the knowledge of how each car interacts with their keys.

Recognising this, a decentralised trust anchor is not the service that makes the token interact with applications - the trust anchor is the token.

Premise 2: Using Token for trust anchor Allows for Layered Design

The second premise for tokens serving as trust anchors is rooted in the layered design of the Internet. This approach has been instrumental to the Internet's success, as it allows for competition among various applications like FTP, USENET,

and the Web. The layered design facilitates the evolution of the Internet, ensuring that even if early protocols fail, more successful ones can be developed without overhauling the foundational layers.

Applying this layered design approach to our car-key token example, we find that platforms like Google Wallet could potentially provide additional authorization features useful to a car wash company. However, the real innovation lies in the hands of car vendors, who could enhance their car-key tokens with functionalities like a car wash mode or geo-fenced keys that restrict the car from being driven onto roads.

If platforms like Google Wallet dictate the features of the tokens, we risk returning to a scenario where the entire Web 2.0 market is waiting for trust anchors to provide specific services before they can innovate. A layered design, where tokens provide services independently of platforms, offers a significant evolutionary advantage.

However, using tokens as trust anchors could lead to services not being sufficiently unified. Two potential solutions can address this issue:

Firstly, industry bodies, Decentralized Autonomous Organizations (DAOs), or the applications themselves can facilitate the development of specifications and standards that guide how tokens should be programmed and interact with various applications. It's worth noting that public-blockchain based standardization process such as ERC is insufficient for standardising trust anchor tokens, the cause for that is to be elaborated in the next section.

Secondly, software libraries could serve as software that abstracts the tokens, similar to how Google Wallet presents a unified interface for different payment cards. Since the tokens themselves already serve as trust anchors, these libraries can focus on facilitating seamless interactions between applications and tokens without needing to externally validate the trustworthiness of the tokens, therefore can be internal components of token-using websites.

In conclusion, tokens, not platforms, should serve as trust anchors. They are the origin of trust and need to be decoupled from token-serving platforms to foster competition and drive innovation, echoing the layered approach that has benefited the Internet. Therefore, efforts to decentralize trust anchors should enrich tokens so they can carry out the functions of trust anchors, rather than merely focusing on decentralizing token-providing platforms. This perspective leads us to the core concept of this paper: Smart Tokens.

Transitioning from Smart Contract Tokens to Smart Tokens

Building on the previous discussion, Smart Contract Tokens, defined in public blockchains such as Ethereum, exhibit certain characteristics that make them a suitable foundation for trust anchors.

Firstly, Smart Contract Tokens' independence from third-party services enhances

their reliability and availability. This is a critical feature for trust anchors, as their role involves facilitating a wide range of web applications. Any downtime or service interruption could have widespread consequences, as Google outages demonstrated[4]. In contrast, blockchain systems such as Ethereum and Bitcoin have exhibited continuous availability, with the unavailability of Smart Contract Tokens historically tied only to specific smart contracts themselves.

Secondly, the deterministic behavior of Smart Contract Tokens, governed by their embedded protocols, ensures consistent and predictable operation. This predictability is crucial for trust anchors, as web applications depending on them need to be confident that the tokens will behave as expected. Decentralised mechanism such as Decentralized Autonomous Organizations (DAOs) enhanced the ability to update smart contracts while adhering to established trust protocols.

However, while Smart Contract Tokens form a solid base, they fall short in the role of trust anchors.

Firstly, Smart Contract Tokens are not inherently designed to function as web services. Although the execution of smart contract function calls, facilitated via Remote Procedure Call (RPC) interfaces, bears a resemblance to the execution of web service function calls through RESTful APIs, they lack a comprehensive service matrix. This includes aspects such as uptime, Service Level Objectives (SLOs), optimisation, load balancing, and confidentiality (ensuring that the RPC server does not leak information about executed calls). Furthermore, the lack of modularity in smart contract tokens is a significant limitation. While web services can be updated to reflect the evolving needs of websites, smart contract calls under the RPC interface cannot be updated without altering the smart contract, which governs its core logic. Thus, using Smart Contract Tokens as trust anchors implies a rigid coupling between token logic and service logic, analogous to the impractical use of SQL as the data language in web front-ends, binding database structure with web interface.

Secondly, Smart Contract Tokens lack a suitable runtime environment. At first glance, it might appear that all Smart Contract Tokens have a runtime environment. For instance, in Ethereum, this would be the Ethereum Virtual Machine (EVM). However, the EVM was specifically designed with the maintenance of states in mind, which is fundamental for value transfer, but not necessarily for functioning as a web service. For instance, it does not maintain the status of a web session, only the states in the blockchain; its operating environment does not demand integrity - if a node returns a false negative in a view function call, there are no repercussions for the node. Moreover, it lacks facilities such as HTTP GET/POST requests and data storage, which are essential for a web environment. Finally, an EVM instance describes smart contract status, not a token instance - there is no EVM to describe a token without also maintaining

[4]For instance, when Google experienced outages three times in 2020 and twice in 2022, it left passengers stranded at airports as airlines required Google login to display their tickets

the entire contract state. This highlights the shortcomings of the EVM when serving as the token runtime environment, which will be explored further in a subsequent section.

Recognising these limitations, it becomes evident that Smart Contract Tokens need to be further developed to serve as effective trust anchors. This enhanced version, "Smart Tokens," was first termed by Virgil Griffith. Griffith coined this term during a casual conversation in a bar in Surry Hills, Sydney, in 2018, as the team behind this paper deliberated on the potential building blocks of Web3. At that point, Web3 was still in its budding state and had not yet been defined as an 'Internet of Value.' The phrase "Smart Token" caught on, and the team adopted the moniker "Smart Token Labs." However, the exact reasoning behind Griffith's nomenclature remains a mystery, as he was unfortunately incarcerated before he could elaborate on it. Hence, we are left to speculate why the envisioned building block for Web3 was christened as a "Smart Token."

The ensuing sections will delve deeper into the concept of Smart Tokens, exploring their design requirements and potential applications, and how they could serve as the building blocks for the next-generation Web.

Smart Token Architecture

Part of the architecture of a Smart Token can be inferred from traditional trust anchors, such as Google Wallet. However, due to the decision to base the Smart Token on public blockchain systems, several significant changes are necessary. Most importantly, Token Runtime Environment replaces the role of Google as the operating environment.

Token Runtime Environment In the Smart Token architecture, the role of smart contract is to ensure the trusted behaviour of the token, such as transfer, creation and destruction of the tokens. A significant part of the trust anchor logic has to be written separately from the smart contract and run in a Token Runtime Environment. This allows Token to fill the role of Trust Anchor.

We enumerate the features needed for such a runtime environment.

Requirements of Token Runtime Environment

Token's Service Interface To support use-cases of web applications, tokens need to provide interfaces for web services. In the example of a car-key token, the website might solicit the colour of the car for identification purpose. Similarly, a smart flight ticket token might provide a call back interface to a hotel website, to inform the website if there is any delay in the passenger's arrival. It's worth noting that these service interfaces do not equal to their smart contract interfaces, as their smart contract is focused on ensuring token rules being followed, and not necessary provision of the services.

Management of Token's State This includes the token's temporary state not associated with the strong, smart contract-based rules. There are common types of states, such as the signed parts of a multi-signature transactions, as well as token-specific states, for instance, in a car-key token, this might include emphemeral car-key authorisations previously generated.

Separation of runtime environment It is vital that the token's runtime code does not share the same memory as the web application it supports. Trust anchors command a high level of trust that other web applications depend on. Without a separate runtime environment for the token and the website using it - such as packing token code with Webpack[5] - users are forced to only use web applications they fully trust, which presents the same trust barrier discussed earlier.

Ability to Verify Code Executed in the Runtime Environment To allow users to view the token as a trust anchor, it must originate from the same source as the token issuer itself, or a source of equivalent trust. The signature on the code running in the secure environment must be traceable to the same source as the smart contract from which the token is derived.

Event-Driven Design The primary case for an event-driven design is to synchronise the status of the token with the blockchain status. Moreover, the token's status and function may be affected by other related tokens, necessitating a cascade of event processing. Furthermore, there are use-cases where the issuer needs an event-based communication channel with the users, such as prompting for token expiry. This necessitates a declarative interface on what event is interesting to the token, as the token code needs to be woken up from a hibernating status[6] to respond to an event.

[5]A common misconception is that packaging technology, such as webpack, can substitute for the runtime environment. However, webpack is a static bundling tool and does not manage runtime processes. Consider a scenario where a website executes trust anchor code, packaged in webpack, within the same environment as the webpage itself. In this case, a car-key token is used on a website. Before any authorisation is made, in order to render a user interface for car selection, the website would already have access to the car's existence, basic information, and the user's relevant key needed for signing. This is because these details are rendered in the webpage's own memory. Such a design, lacking separation between the runtime environment of the token and the website using it, compels the user to only use web applications they fully trust. This reintroduces the same trust barrier previously discussed, thereby highlighting the pivotal role of a distinct runtime environment for token operations.

[6]Unlike browser JavaScript, the event-driven design of a token cannot assume that the token's code is actively running when an event occurs, especially as the user may not have a browser window open at that time. This necessitates the need for a declarative interface that specifies which events are relevant to the token. Without such an interface, the token code would need to be continually awakened from a dormant state to respond to numerous events, many of which may be irrelevant, leading to unnecessary computational effort and inefficient operation. This declarative interface, therefore, serves as a filter, allowing the token code to remain in a hibernation-like state until a relevant event requires its attention.

Composability In the absence of a platform, tokens need to rely on standards and composable interfaces to work with each other. This is straightforward in the case of composable tokens from the same team. However, this cannot be guaranteed[7]. Again, declarative components are needed here as when the use-case arises, a system needs to instantiate the relevant tokens without first running them in memory.

Security Given that a token can enable multiple web applications, it often becomes a more attractive target for attacks than the website itself. The design of the token runtime must take this into account. Furthermore, to safeguard websites from generating incorrect or malicious transactions, transactions should be generated inside the Token's Runtime Environment, in response to the need of the web application it supports. For example, a car cleaning service website shouldn't have the code to generate a car-key authorisation; it should only have the code to call for such an authorisation to be generated.

Token's User Interface The token's user interface is a crucial element of the token runtime code. It is essential for the token to create its own user interface to elevate the trust to the website to meet the web application's demand. The token's visual presentation should be created in a secured Token Runtime Environment. This is akin to how Metamask, a browser plugin, provides the interface between the user and a blockchain-enabled Dapp for connecting the wallet and authorising transactions. However, in the case of smart tokens, the UI needs to be specific to the token to enable its use-cases[8].

The user interface (UI) of a token is a crucial aspect of the token runtime code. It must be tailored to the token to enable its specific use-cases. For instance, the UI for a user to authorise a car-key token for a car cleaning service would be entirely different from another user, who holds a smart-flight ticket (also a smart token), booking a connected hotel stay. Even with the same token, different use-actions necessitate different user interfaces. Providing the location through a car-key token is a simpler user interface than authorising a website to send staff to drive it. The user interface of a token is determined by the actions it can perform, which web applications depend on.

One argument against the use of token-specific interfaces suggests that websites tend to customise everything within their operational environment. However, the consistent visual presentation of Google Pay and Google Login across different

[7]In instances where two tokens are closely related, such as the car-key token and the car-ownership token, their creation process is likely to be closely linked due to shared developers or teams. However, this close collaboration cannot be universally guaranteed. For instance, if a user wishes to loan their car to a website for an extended period of time, beyond a simple cleaning service, the car's registration plate - another smart token - becomes a dependency. This token would likely originate from a government entity, demonstrating that token dependencies can extend beyond immediate development teams and encompass larger regulatory bodies.

[8]Say for example, if Tesla Car-Key is a smart token, users need to see Tesla taking over the process and pass authorisation to the website, similar to how Google Wallet or MetaMask taking over the process and passing the result to the website.

websites of various colours and fonts demonstrates that the trust factor - users recognising a symbol of trust to carry out actions - outweighs the visual experience alone.

A related argument proposes that the token interface could be supplied by a website library, rather than being generated by the token's secure runtime environment. This reflects a misunderstanding of the dual role of trust anchors, in that they provide both function and trust. The token user interface on a website provides no more trust than the website itself, as it is entirely under the website's control. Therefore, the token-specific user interface must be generated within the token's secure runtime environment to ensure trust is maintained.

In conclusion, the design choices for a smart token's runtime environment for serving as a trust anchor can be generalised in a few terms. It is a separate runtime environment from the application it supports, is event-driven, and has declarative components. This design ensures that the token can serve as a trust anchor and provide the necessary functions while maintaining a high level of security.

A Tripartite Approach

The previous section delineated the prerequisites for Smart Tokens to function as trust anchors. The team behind this paper has developed two technologies to address these requirements.

1. TokenScript TokenScript is a scripting framework that amalgamates the token code, destined to execute in the Token Runtime Environment, with the declarative components necessary for the tokens to operate as trust anchors. These components include events, mechanisms for synchronising with blockchain token status, and definitions of actions that websites can invoke.

At a high level, TokenScript is an XML container that combines declarations to trigger, render, and correctly manage the code, along with the code itself, into a single, signed, deployable format. It further employs XML signatures to permit the publication of TokenScript segments that are sufficiently small to support a single use-case and modular enough to swap in and out data segments. This could include changing the language of the token user interface based on the use-case.

At a more granular level, TokenScript maintains, in a declarative manner, the relationships between tokens that can function together. This includes potential modules from industry bodies for standardising sections of token behaviour, such as a standard lock-unlock function for all car-key tokens, and commands the attestation formats it can accept and the list of websites it can safely access for its dependency data. The details are necessary to fend off various forms of attack, prevent resource abuse, and inform the system that utilises it of its states.

TokenScript, a recommended smart token standard work in progress, is currently hosted under OASIS-open for standardisation. An earlier paper detailing its design can be found at https://github.com/TokenScript/documents/releases.

2. ERC 5169 Family of ERCs This component of the tech stack is a protocol linking smart contracts with the Smart Token's TokenScript. There are several ERCs for different usage scenarios. For example, ERC 5170 utilises a separate TokenScript signing key from the smart contract signing key, allowing for greater flexibility, role separation, and frequent updates to TokenScript without the need for frequent updates to Smart Contracts.

3. Smart Layer Network The Smart Layer Network is the Token Runtime Environment designed to enable Smart Tokens through the execution of TokenScripts. In simpler terms, it's a container that runs TokenScripts, akin to a Kubernetes Engine that runs instances of tokens, similar to how Google Wallet is a container running their own tokens. This network provides RESTful APIs for the Smart Token-based Trust Anchors.

However, the actual implementation is more complex as it is designed to operate as a decentralised service network. The enforcement of Service Level Agreements, the mechanism to ensure the network is ready to instantiate specific types of tokens, to reward token hosting nodes, as well as load balancing and token/node assignment are specific challenges that the design needs to address. The Smart Layer Network has its own paper available at https://github.com/TokenScript/documents/releases.

Due to the scope of this paper, we will not delve into the details of these technologies.

Wood, Gavin. "The Future of the Decentralized Web," 2017.